Policy Languages

• Express security policies in a precise way
• High-level languages
  – Policy constraints expressed abstractly
• Low-level languages
  – Policy constraints expressed in terms of program options, input, or specific characteristics of entities on system
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: Web Browser

- Goal: restrict actions of Java programs that are downloaded and executed under control of web browser
- Language specific to Java programs
- Expresses constraints as conditions restricting invocation of entities
Expressing Constraints

- **Entities are classes, methods**
  - Class: set of objects that an access constraint constrains
  - Method: set of ways an operation can be invoked

- **Operations**
  - Instantiation: $s$ creates instance of class $c$: $s \rightarrow c$
  - Invocation: $s1$ executes object $s2$: $s1 \rightarrow s2$

- **Access constraints**
  - \texttt{deny}(s op x) \texttt{when} b
  - While $b$ is true, subject $s$ cannot perform $op$ on (subject or class) $x$; empty $s$ means all subjects
DTEL

- Basis: access can be constrained by types
- Combines elements of low-level, high-level policy languages
  - Implementation-level constructs express constraints in terms of language types
  - Constructs do not express arguments or inputs to specific system commands
Example

- Goal: users cannot write to system binaries
- Subjects in administrative domain can
  - User must authenticate to enter that domain
- Subjects belong to domains:
  - \textit{d\_user} ordinary users
  - \textit{d\_admin} administrative users
  - \textit{d\_login} for login
  - \textit{d\_daemon} system daemons
Types

- **Object types:**
  - $t_{sysbin}$ executable system files
  - $t_{readable}$ readable files
  - $t_{writable}$ writable files
  - $t_{dte}$ data used by enforcement mechanisms
  - $t_{generic}$ data generated from user processes

- **For example, treat these as partitions**
  - In practice, files can be readable and writable; ignore this for the example
Domain Representation

- Sequence
  - First component is list of programs that start in the domain
  - Other components describe rights subject in domain has over objects of a type
    (crwd->t_writable)
    means subject can create, read, write, and list (search) any object of type t_writable
\textit{d\_daemon} Domain

\begin{verbatim}
domain d_daemon = (/sbin/init),
    (crwd->t_writable),
    (rd->t_generic, t_readable, t_dte),
    (rxd->t_sysbin),
    (auto->d_login);
\end{verbatim}

- Compromising subject in \textit{d\_daemon} domain does not enable attacker to alter system files
  - Subjects here have no write access
- When /sbin/init invokes login program, login program transitions into \textit{d\_login} domain
d_admin Domain

domain d_admin =
    (/usr/bin/sh, /usr/bin/csh, /usr/bin/ksh),
    (crwxd->t_generic),
    (crwxd->t_readable, t_writable, t_dte,
    t_sysbin),
    (sigtstp->d_daemon);

• sigtstp allows subjects to suspend processes in d_daemon domain

• Admin users use a standard command interpreter
\textit{d\_user} Domain

\begin{verbatim}
domain d_user =
    (/usr/bin/sh, /usr/bin/csh, /usr/bin/ksh),
    (crwxd->tGeneric),
    (rxd->t_sysbin),
    (crwd->t_writable),
    (rd->t_readable, t_dte);
\end{verbatim}

- No auto component as no user commands transition out of it
- Users cannot write to system binaries
**`d_login` Domain**

```plaintext
domain d_login =
    (/usr/bin/login),
    (crwd->t_writable),
    (rd->t_readable, t_generic, t_dte),
    setauth,
    (exec->d_user, d_admin);
```

- Cannot execute anything except the transition
  - Only `/usr/bin/login` in this domain
- `setauth` enables subject to change UID
- `exec` access to `d_user`, `d_admin` domains
Set Up

initial_domain = d_daemon;

- System starts in *d_daemon* domain
assign –r t_generic /;
assign –r t_writable /usr/var, /dev, /tmp;
assign –r t_readable /etc;
assign –r –s dte_t /dte;
assign –r –s t_sysbin /sbin, /bin,
    /usr/bin, /usr/sbin;

- These assign initial types to objects
- –r recursively assigns type
- –s binds type to name of object (delete it, recreate it, still of given type)
Add Log Type

- Goal: users can’t modify system logs; only subjects in \textit{d_admin}, new \textit{d_log} domains can type \textit{t_readable}, \textit{t_writable}, \textit{t_sysbin}, \textit{t_dte}, \textit{t_generic}, \textit{t_log};

- New type \textit{t_log}

\begin{verbatim}
domain d_log =
(/usr/sbin/syslogd),
(crwd->t_log),
(rwd->t_writable),
(rd->t_generic, t_readable);
\end{verbatim}

- New domain \textit{d_log}
Fix Domain and Set-Up

domain d_daemon = (/sbin/init),
(crwd->t_writable),
(rxd->t_readable),
(rd->t_generic, t_dte, t_sysbin),
(auto->d_login, d_log);

• Subject in d_daemon can invoke logging process
  – Can log, but not execute anything
assign -r t_log /usr/var/log;
assign t_writable /usr/var/log/wtmp,
   /usr/var/log/utmp;

• Set type of logs
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
    \texttt{xhost +groucho -chico}
- Connections from host groucho allowed
- Connections from host chico not allowed
Example: tripwire

- File scanner that reports changes to file system and file attributes
  - `tw.config` describes what may change
  
  ```
  /usr/mab/tripwire +gimnpsu012345678-a
  ```

- Check everything but time of last access ("-a")
  - database holds previous values of attributes
Example Database Record

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

• System administrators not expected to edit database to set attributes properly
• Checking for changes with tripwire is easy
  – Just run once to create the database, run again to check
• Checking for conformance to policy is harder
  – Need to either edit database file, or (better) set system up to conform to policy, then run tripwire to construct database
Example English Policy

• Computer security policy for academic institution
  – Institution has multiple campuses, administered from central office
  – Each campus has its own administration, and unique aspects and needs
• Authorized Use Policy
• Electronic Mail Policy
Authorized Use Policy

- Intended for one campus (Davis) only
- Goals of campus computing
  - Underlying intent
- Procedural enforcement mechanisms
  - Warnings
  - Denial of computer access
  - Disciplinary action up to and including expulsion
- Written informally, aimed at user community
Electronic Mail Policy

• Systemwide, not just one campus
• Three parts
  – Summary
  – Full policy
  – Interpretation at the campus
Summary

• Warns that electronic mail not private
  – Can be read during normal system administration
  – Can be forged, altered, and forwarded

• Unusual because the policy alerts users to the threats
  – Usually, policies say how to prevent problems, but do not define the threats
Summary

• What users should and should not do
  – Think before you send
  – Be courteous, respectful of others
  – Don’t interfere with others’ use of email

• Personal use okay, provided overhead minimal

• Who it applies to
  – Problem is UC is quasi-governmental, so is bound by rules that private companies may not be
  – Educational mission also affects application
Full Policy

• Context
  – Does not apply to Dept. of Energy labs run by the university
  – Does not apply to printed copies of email
    • Other policies apply here

• E-mail, infrastructure are university property
  – Principles of academic freedom, freedom of speech apply
  – Access without user’s permission requires approval of vice chancellor of campus or vice president of UC
  – If infeasible, must get permission retroactively
Uses of E-mail

• Anonymity allowed
  – Provided it doesn’t break laws or other policies
• Can’t interfere with others’ use of e-mail
  – No spam, letter bombs, e-mailed worms, etc.
• Personal e-mail allowed within limits
  – Cannot interfere with university business
  – Such e-mail may be a “university record” subject to disclosure
Security of E-mail

• University can read e-mail
  – Won’t go out of its way to do so
  – Allowed for legitimate business purposes
  – Allowed to keep e-mail robust, reliable

• Archiving and retention allowed
  – May be able to recover e-mail from end system
    (backed up, for example)
Implementation

• Adds campus-specific requirements and procedures
  – Example: “incidental personal use” not allowed if it benefits a non-university organization
  – Allows implementation to take into account differences between campuses, such as self-governance by Academic Senate

• Procedures for inspecting, monitoring, disclosing e-mail contents

• Backups
Confidentiality Policy

• Goal: prevent the unauthorized disclosure of information
  – Deals with information flow
  – Integrity incidental

• Multi-level security models are best-known examples
  – Bell-LaPadula Model basis for many, or most, of these
Bell-LaPadula Model, Step 1

- Security levels arranged in linear ordering
  - Top Secret: highest
  - Secret
  - Confidential
  - Unclassified: lowest

- Levels consist of *security clearance* $L(s)$
  - Objects have *security classification* $L(o)$
## Example

<table>
<thead>
<tr>
<th>security level</th>
<th>subject</th>
<th>object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top Secret</td>
<td>Tamara</td>
<td>Personnel Files</td>
</tr>
<tr>
<td>Secret</td>
<td>Samuel</td>
<td>E-Mail Files</td>
</tr>
<tr>
<td>Confidential</td>
<td>Claire</td>
<td>Activity Logs</td>
</tr>
<tr>
<td>Unclassified</td>
<td>Ulaleyl</td>
<td>Telephone Lists</td>
</tr>
</tbody>
</table>

- Tamara can read all files
- Claire cannot read Personnel or E-Mail Files
- Ulaleyl can only read Telephone Lists
Reading Information

- Information flows *up*, not *down*
  - “Reads up” disallowed, “reads down” allowed
- Simple Security Condition (Step 1)
  - Subject $s$ can read object $o$ iff, $L(o) \leq L(s)$ and $s$ has permission to read $o$
    - Note: combines mandatory control (relationship of security levels) and discretionary control (the required permission)
  - Sometimes called “no reads up” rule
Writing Information

• Information flows up, not down
  – “Writes up” allowed, “writes down” disallowed

• *-Property (Step 1)
  – Subject \( s \) can write object \( o \) iff \( L(s) \leq L(o) \) and \( s \) has permission to write \( o \)
    • Note: combines mandatory control (relationship of security levels) and discretionary control (the required permission)
  – Sometimes called “no writes down” rule