Chapter 4: Security Policies

• Overview
• The nature of policies
  – What they cover
  – Policy languages
• The nature of mechanisms
  – Types
  – Secure vs. precise
• Underlying both
  – Trust
Overview

• Overview
• Policies
• Trust
• Nature of Security Mechanisms
• Policy Expression Languages
• Limits on Secure and Precise Mechanisms
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 to which a particular access constraint is applied.
  – Method: set of ways an operation can be invoked

• Operations
  – Instantiation: $s$ creates instance of class $c$: $s \rightarrow c$
  – Invocation: $s_1$ executes object $s_2$: $s_1 \rightarrow s_2$

• Access constraints
  – deny($s$ op $x$) when $b$
  – While $b$ is true, subject $s$ cannot perform $op$ on (subject or class) $x$; empty $s$ means all subjects
Sample Constraint

- At most 100 network connections open
- *Socket* class defines network interface
  - `Network.numconns` gives number of active network connections
- Constraint
  
  `deny(-| Socket) when (Network.numconns >= 100)`
DTEL (Domain-type enforcement language)

• Basis: access can be constrained by types
• Each object is associated with a type
• Each subject is associated with a domain
• The constructs of the language constrain the actions that a member of a domain can perform on an object of a specific type.
  – A subject cannot execute a text file, but it can execute an object file.
Example

- **Goal:** users cannot write to system binaries
- **Subjects in administrative domain can**
  - User must authenticate to enter that domain
- **Subjects belong to domains:**
  - `d_user` ordinary users
  - `d_admin` administrative users
  - `d_login` for login
  - `d_daemon` system daemons
- **login** program (`d_login` domain) controls access between `d_user` and `d_admin`. 

*Computer Security: Art and Science*
©2002-2004 Matt Bishop
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
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
    
    \[(\text{crwd}\rightarrow\text{t}\_\text{writable})\]

    means subject can create, read, write, and list (search) any object of type \(\text{t}\_\text{writable}\).
$d_{daemon}$ Domain

- When the *init* program begins, it starts in this domain.
  - It can create(c), read(r), write(w), and do a directory search(d) of any object of type *t_writable*.
  - It can read, search, and execute(x) any object of type *t_sysbin*
  - It can read and search anything of type *t_generic*, *t_readable*, or *t_dte*.
  - When the *init* program invokes the login program, the login program transitions to the *d_login* domain automatically.
d_daemon Domain

domain d_daemon = (/sbin/init),
    (crwd->t_writable),
    (rd->t_generic, t_readable, t_dte),
    (rxd->t_sysbin),
    (auto->d_login);
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)
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
  - Provides a language for controlling access to the console

- 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
**Example: tripwire**

(ftp://ftp.cerias.purdue.edu/pub/tools/unix/ids/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

- Tripwire assumes a policy of constancy
  - Records an initial state and on subsequent runs reports files whose settings have changed.
Example Database Record

/usr/mab/tripwire/README 0 ..../. 100600 45763 1 917 10 33242 .gtPvf .gtPvY .gtPvY 0 .ZD4cc0Wr8i21ZKaI..LUOr3 .0fwo5:hf4e4.8TAgdOV4ubv ?....... ...9b3 1M4GX01xbGIX0oVuGo1h15z3 ?:Y9jfa04rdzM1q:eqt1APgHk ?.Eb9yo.2zkEh1XKovX1:d0wF0kfAvC ?1M4GX01xbGIX2947jdyrrior38h15z3 0

- 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: The RIACS file system checker

- Emphasizes the ability to set policy and then check policy and then check for conformance.
- Uses a database file and records fixed attributes.
  - /etc/pac 0755 1 root root 16384 12 22341 Jan 12, 1999 at 12:47:54
- After generating such a file, the analyst can change the values as appropriate.
  - The next run, file system state is compared with these values.