Overview

- Policies
- Trust
- Nature of Security Mechanisms
- Policy Expression Languages
Security Policy

- Policy partitions system states into:
  - Authorized (secure)
    - These are states the system can enter
  - Unauthorized (nonsecure)
    - If the system enters any of these states, it’s a security violation

- Secure system
  - Starts in authorized state
  - Never enters unauthorized state
Confidentiality

• $X$ set of entities, $I$ information
• $I$ has the confidentiality property with respect to $X$ if no $x \in X$ can obtain information from $I$
• $I$ can be disclosed to others
• Example:
  – $X$ set of students
  – $I$ final exam answer key
  – $I$ is confidential with respect to $X$ if students cannot obtain final exam answer key
Integrity

• $X$ set of entities, $I$ information

• $I$ has the *integrity* property with respect to $X$ if all $x \in X$ trust information in $I$

• Types of integrity:
  – Trust $I$, its conveyance and protection (data integrity)
  – $I$ information about origin of something or an identity (origin integrity, authentication)
  – $I$ resource: means resource functions as it should (assurance)
Availability

- $X$ set of entities, $I$ resource
- $I$ has the *availability* property with respect to $X$ if all $x \in X$ can access $I$
- Types of availability:
  - Traditional: $x$ gets access or not
  - Quality of service: promised a level of access (for example, a specific level of bandwidth) and not meet it, even though some access is achieved
Policy Models

• Abstract description of a policy or class of policies
• Focus on points of interest in policies
  – Security levels in multilevel security models
  – Separation of duty in Clark-Wilson model
  – Conflict of interest in Chinese Wall model
Mechanisms

• Entity or procedure that enforces some part of the security policy
  – Access controls (like bits to prevent someone from reading a homework file)
  – Disallowing people from bringing CDs and floppy disks into a computer facility to control what is placed on systems
Question

- Policy disallows cheating
  - Includes copying homework, with or without permission
- CS class has students do homework on computer
- Anne forgets to read-protect her homework file
- Bill copies it
- Who cheated?
  - Anne, Bill, or both?
Answer Part 1

• Bill cheated
  – Policy forbids copying homework assignment
  – Bill did it
  – System entered unauthorized state (Bill having a copy of Anne’s assignment)

• If not explicit in computer security policy, certainly implicit
  – Not credible that a unit of the university allows something that the university as a whole forbids, unless the unit explicitly says so
Answer Part #2

• Anne didn’t protect her homework
  – Not required by security policy
• She didn’t breach security
• If policy said students had to read-protect homework files, then Anne did breach security
  – She didn’t do this
Types of Security Policies

- Military (governmental) security policy
  - Policy primarily protecting confidentiality
- Commercial security policy
  - Policy primarily protecting integrity
- Confidentiality policy
  - Policy protecting only confidentiality
- Integrity policy
  - Policy protecting only integrity
Integrity and Transactions

• Begin in consistent state
  – “Consistent” defined by specification

• Perform series of actions (transaction)
  – Actions cannot be interrupted
  – If actions complete, system in consistent state
  – If actions do not complete, system reverts to beginning (consistent) state
Trust

Administrator installs patch

1. Trusts patch came from vendor, not tampered with in transit
2. Trusts vendor tested patch thoroughly
3. Trusts vendor’s test environment corresponds to local environment
4. Trusts patch is installed correctly
Trust in Formal Verification

• Gives formal mathematical proof that given input $i$, program $P$ produces output $o$ as specified

• Suppose a security-related program $S$ formally verified to work with operating system $O$

• What are the assumptions?
Trust in Formal Methods

1. Proof has no errors
   - Bugs in automated theorem provers
2. Preconditions hold in environment in which $S$ is to be used
3. $S$ transformed into executable $S'$ whose actions follow source code
   - Compiler bugs, linker/loader/library problems
4. Hardware executes $S'$ as intended
   - Hardware bugs (Pentium $\pm 00\pm$ bug, for example)
Types of Access Control

• Discretionary Access Control (DAC, IBAC)
  – Individual user sets access control mechanism to allow or deny access to an object

• Mandatory Access Control (MAC)
  – System mechanism controls access to object, and individual cannot alter that access

• Originator Controlled Access Control (ORCON)
  – Originator (creator) of information controls who can access information
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: 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

```c
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 {
  grantees /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

```javascript
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 3rd login failure, net security admins will disable account and log event

```java
inst oblig loginFailure {
    on loginfail(userid, 3);
    subject s=/NetAdmins/SecAdmins;
    target t=/NetAdmins/users ^ (userid);
    do t.disable() -> s.log(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: 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

January 22, 2016
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