Security Policies

COMP620

What is a Security Policy?

- A security policy defines “secure” for a system
- How do you know that you system is secure until you define what secure is?
- A security policy defines what a user can or cannot do

Policy Source

Security policies can be created by:

- Organizations (corporations or universities)
- Government
- Software vendors

Who Cares?

- Even if you have the most secure encryption system, it will be of little use if you give everyone the keys

Confidentiality

- *X* set of entities, *I* information
- *I* has *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 *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 *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

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

An administrator who installs an OS 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*

Trust in Formal Methods

1. Proof has no errors *(can every COMP681 student do this?)*
   - 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 `f00f` 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

Question ?

- A university policy disallows cheating which is defined to include copying another student's homework (with or without permission)
- CS class has students do homework on lab computers
- Alice forgets to read-protect her homework file
- Bob copies it
- Who cheated?
  - Alice, Bob , or both?

Answer Part 1

- Bob cheated
  - Policy forbids copying homework assignment
  - Bob did it
  - System entered unauthorized state (Bob 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

- Alice 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 Alice did breach security
  - She didn't do this

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

Mechanisms

Mechanisms: may be

- technical, in which controls in the computer enforce the policy; for example, the requirement that a user supply a password to authenticate herself before using the computer
- procedural, in which controls outside the system enforce the policy; for example, firing someone for ringing in a disk containing a game program obtained from an untrusted source

Policy Language

Policy may be expressed in

- natural language, which is usually imprecise but easy to understand;
- mathematics, which is usually precise but hard to understand;
- policy languages, which look like some form of programming language and try to balance precision with ease of understanding

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: Java in a 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

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