# Security Policies

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

1

# Security Policy

- Policy partitions system states into:
  - Authorized (secure)
    - These are states the system can enter
  - Unauthorized (non-secure)
    - If the system enters any of these states, a breach of security has occurred
- Secure system
  - Starts in an authorized state
  - Never enters an unauthorized state

2

## Question

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

3

## 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)
- Department's responsibility?
  - 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

4

## Answer Part 2

- Anne did not protect her homework
  - Not required by security policy
- She did not breach security
- If policy said students had to read-protect homework files, then Anne did breach security
  - Because she did not do this

5

## Mechanisms

- Entity or procedure that enforces some part of the security policy
  - Access controls (set 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

6

# Policy Models

- Abstract description of a policy or class of policies
- Focus on points of interest in policies
  - Confidentiality Policies
    - Security levels in multilevel security models
    - Prohibit direct or indirect information flow
  - Integrity Policies
    - Separation of duty in Clark-Wilson model
    - Restrict who/how data can be modified
    - Conflict of interest in Chinese Wall model (both conf./int.)
  - Availability Policies
    - describe what type/level of service must be provided

7

# Integrity and Transactions

Some integrity policies use notion of transaction

- Begin in consistent state
  - "Consistent" defined by specification (e.g., database)
- 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
  - Note: two-phase commitment is not a security mechanism

8

## (some) Types of Access Control

- Discretionary Access Control (DAC, IBAC)
  - individual user sets access control mechanism to allow or deny access to an object; based on identity
- 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
  - Own right not enough to control rights

9

## 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

10

## High-Level Policy Languages

- Constraints expressed independently of enforcement mechanism
- Constraints restrict actions and entities
- Constraints expressed unambiguously
  - Requires a precise language, usually a mathematical, logical, or programming-like language; English typically not precise enough

11

## 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 creation of classes and invocation of entities
- Independent of enforcement mechanism

12

# Expressing Constraints

- Entities are classes, methods
  - Class: set of objects that an access constraint constrains (e.g., file, socket)
  - Method: set of ways an operation can be invoked (e.g., file.read() )
- Operations
  - Instantiation: *s* creates instance of class *c*: $s \dashv\mid c$
  - Invocation: $s_1$ executes object $s_2$: $s_1 \mid\!\rightarrow s_2$
- 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

13

# Sample Constraints

- Downloaded program cannot access password file on UNIX system
  - Program's class and methods for files:

```
class File {
  public file(String name);
  public String getfilename();
  public char read();
```

  - Constraint:

```
deny( |-> file.read) when
       (file.getfilename() == "/etc/passwd")
```

- Program cannot open network connection when 100 connections already exist
  - Constraint:

```
deny( |- socket) when (network.numconns >= 100)
```

14

## Low-Level Policy Languages

- Set of inputs or arguments to commands to check or set constraints on system
- Low level of abstraction
  - Need details of system, commands

15

## Example: X Window System

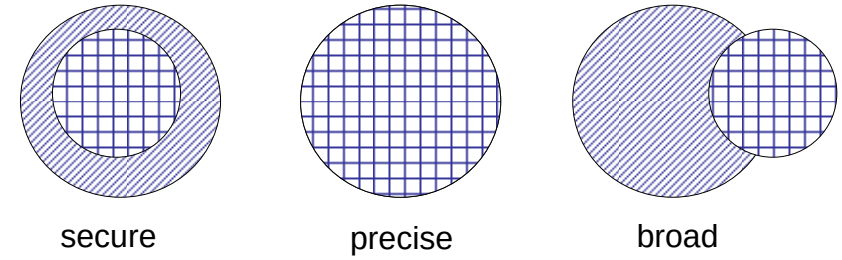
- UNIX-based X11 Windowing System
- Access to X11 display controlled by list
  - List says what hosts allowed, disallowed access

```
xhost +groucho -chico
```

  - Connections from host groucho allowed
  - Connections from host chico not allowed

16

## Types of Mechanisms





17

## Secure, Precise Mechanisms

- Can one devise a procedure for developing a mechanism that is ~~both secure *and*~~ precise?
  - Consider confidentiality policies only here
  - Integrity policies produce same result
- Program = function with multiple inputs and one output
  - a function $p$: $I_1 \times \ldots \times I_n \rightarrow R$ is a program with $n$ inputs $i_k \in I_k$, $1 \leq k \leq n$, and one output $r \in R$

18

## Programs and Postulates

- *Observability Postulate*: the output of a function encodes all available information about its inputs
  - Shared resources could be monitored
  - Covert channels considered part of the output
- Example: authentication function
  - Inputs name, password; output Good or Bad
  - If name invalid, immediately print Bad; else access database
  - If time output of Bad, can determine if name valid
  - This means timing is part of output

19

## Protection Mechanism

- For a function $p : I_1 \times ... \times I_n \rightarrow R$, a protection mechanism is a function $m : I_1 \times ... \times I_n \rightarrow R \cup E$ for which, when $i_k \in I_k$, $1 \le k \le n$, either
  - $m(i_1, ..., i_n) = p(i_1, ..., i_n)$ or
  - $m(i_1, ..., i_n) \in E$.
- $E$ is set of outputs that indicate error/abnormal result
  - In above example, E = { "Password Database Missing", "Password Database Locked" }

20

## Confidentiality Policy

- Confidentiality policy $c$ for program $p$ says which inputs can be revealed
  - Formally, for $p : I_1 \times ... \times I_n \rightarrow R$, it defines a set $A \subseteq I_1 \times ... \times I_n$ of inputs available to observer
- Protection mechanism is function

  $m: I_1 \times ... \times I_n \rightarrow R \cup E$
  - $m$ is *secure* **iff** $\exists\ m' : A \rightarrow R \cup E$ such that, for all $i_k \in I_k$, $1 \leq k \leq n$, $m(i_1, ..., i_n) = m'(a)$ for some a ∈ A

21

## Precision

Security policy may be over-restrictive

- Precision measures how over-restrictive

$m_1$, $m_2$ distinct protection mechanisms for program $p$ under policy $c$

- $m_1$ is as precise as $m_2$ ($m_2 \leq m_1$) if, for all inputs $i_1$, ..., $i_n$, $m_2(i_1, ..., i_n) = p(i_1, ..., i_n) \Rightarrow m_1(i_1, ..., i_n) = p(i_1, ..., i_n)$
- $m_1$ more precise than $m_2$ ($m_1 > m_2$) if <u>in addition</u> there is an input ($i_1'$, ..., $i_n'$) such that $m_1(i_1', ..., i_n') = p(i_1', ..., i_n')$ and $m_2(i_1', ..., i_n') \neq p(i_1', ..., i_n')$.

22

## Combining Mechanisms

- $m_1$, $m_2$ protection mechanisms
- $m_3 = m_1 \cup m_2$
  - For inputs on which $m_1$ or $m_2$ returns same value as $p$, $m_3$ does also; otherwise, $m_3$ returns value of $m_1$
- Theorem: if $m_1$, $m_2$ secure, then $m_3$ secure
  - Also, $m_1 \leq m_3$ and $m_2 \leq m_3$
  - Follows from definitions of secure, precise, and $m_3$

23

## Existence Theorem

- For any program $p$ and security policy $c$, there exists a ~~precise,~~ secure mechanism $m^*$ such that, for all secure mechanisms $m$ associated with $p$ and $c$, $m \leq m^*$
  - Maximally precise mechanism
  - Ensures security
  - Minimizes number of denials of legitimate actions

24

## Lack of Effective Procedure

- There is no effective procedure that determines a maximally precise, secure mechanism for any policy and program.

25