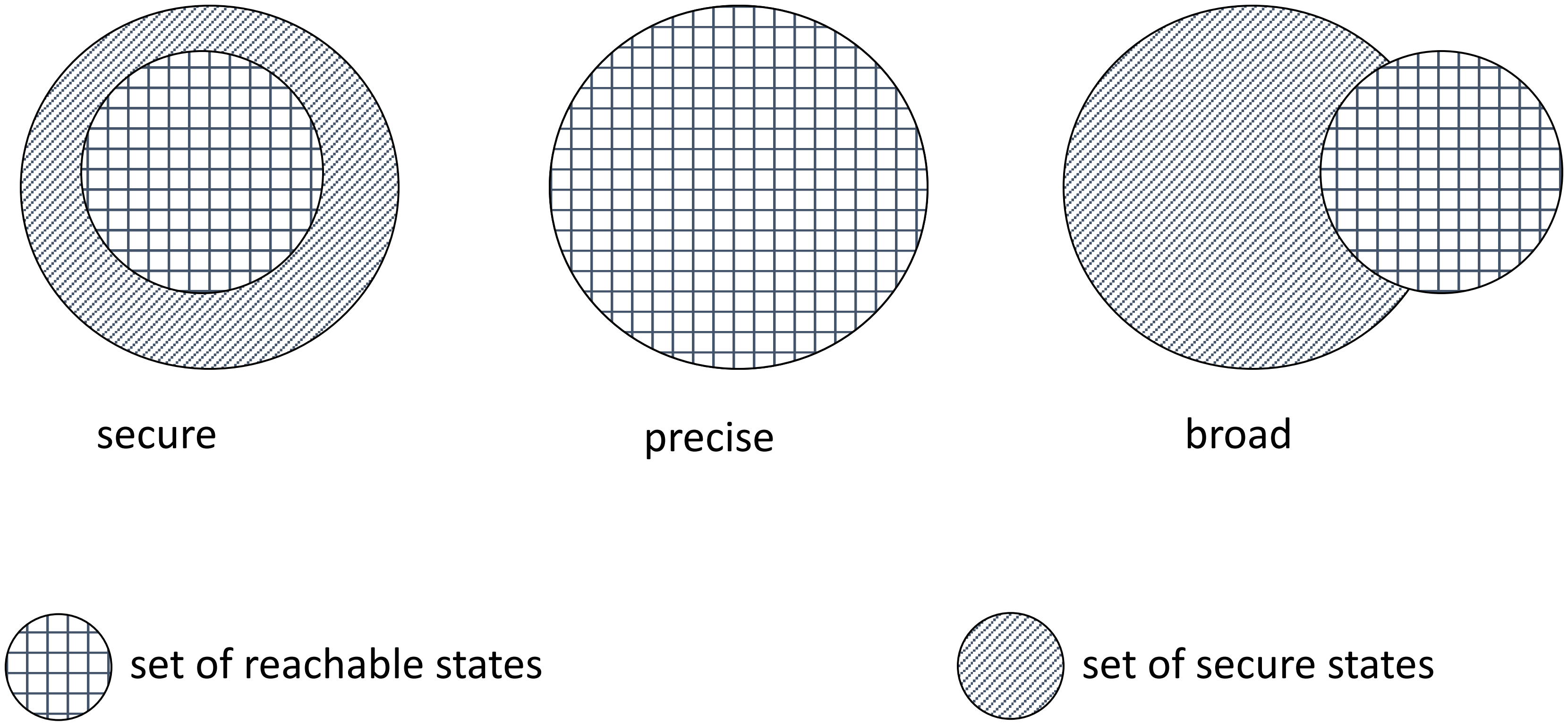
ECS 235B Module 15
Precise and Secure Policies

Types of Mechanisms



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 a function with multiple inputs and one output
  - Let p be a function $p: I_1 \times ... \times I_n \rightarrow R$. Then p is a program with n inputs $i_k \in I_k$, $1 \leq k \leq n$, and one output $r \in R$

Programs and Postulates

- Observability Postulate: the output of a function encodes all available information about its inputs
  - 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
  - Problem: time output of Bad, can determine if name valid
  - This means timing is part of output

Protection Mechanism

- Let p be a function $p: I_1 \times ... \times I_n \rightarrow R$. A *protection mechanism m* is a function

$$m: I_1 \times ... \times I_n \rightarrow R \cup E$$

for which, when $i_k \in I_k$, $1 \leq k \leq n$, either

  - $m(i_1, ..., i_n) = p(i_1, ..., i_n)$ or
  - $m(i_1, ..., i_n) \in E$.
- E is set of error outputs
  - In above example, E = { "Password Database Missing", "Password Database Locked" }

Confidentiality Policy

- Confidentiality policy for program *p* says which inputs can be revealed
  - Formally, for $p: I_1 \times ... \times I_n \rightarrow R$, it is a function $c: I_1 \times ... \times I_n \rightarrow A$, where

    $$A \subseteq I_1 \times ... \times I_n$$
  - *A* is set of inputs available to observer
- Security mechanism is function

$$m: I_1 \times ... \times I_n \rightarrow R \cup E$$

  - *m* is *secure* if and only if $\exists\, m' : A \rightarrow R \cup E$ such that,

    $$\forall i_k \in I_k, 1 \leq k \leq n, m(i_1, ..., i_n) = m'(c(i_1, ..., i_n))$$
  - *m* returns values consistent with *c*

Examples

- $c(i_1, ..., i_n) = C$, a constant
  - Deny observer any information (output does not vary with inputs)
- $c(i_1, ..., i_n) = (i_1, ..., i_n)$, and $m' = m$
  - Allow observer full access to information
- $c(i_1, ..., i_n) = i_1$
  - Allow observer information about first input but no information about other inputs.

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 *as precise as* m_2 ($m_1 \approx m_2$) if, for all inputs $i_1, \ldots, i_n$,
    $m_2(i_1, \ldots, i_n) = p(i_1, \ldots, i_n) \Rightarrow m_1(i_1, \ldots, i_n) = p(i_1, \ldots, i_n)$
  - m_1 *more precise than* m_2 ($m_1 \sim m_2$) if there is an input ($i_1', \ldots, i_n'$) such that $m_1(i_1', \ldots, i_n') = p(i_1', \ldots, i_n')$ and $m_2(i_1', \ldots, i_n') \neq p(i_1', \ldots, i_n')$.

Combining Mechanisms

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

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*, $m^* \approx m$
  - Maximally precise mechanism
  - Ensures security
  - Minimizes number of denials of legitimate actions

Lack of Effective Procedure

- There is no effective procedure that determines a maximally precise, secure mechanism for any policy and program.
  - Sketch of proof: let policy *c* be constant function, and *p* compute function *T*(*x*). Assume *T*(*x*) = 0. Consider program *q*, where

```
z = p;
if z = 0 then y := 1 else y := 2;
halt;
```

Rest of Sketch

- *m* associated with *q*, *y* value of *m*, *z* output of *p* corresponding to $T(x)$
- $\forall x\ [T(x) = 0] \rightarrow m(x) = 1$
- $\exists x'\ [T(x') \neq 0] \rightarrow m(x) = 2$ or $m(x)$ undefined
- If you can determine *m*, you can determine whether $T(x) = 0$ for all *x*
- Determines some information about input (is it 0?)
- Contradicts constancy of *c*.
- Therefore no such procedure exists