IS 2150 / TEL 2810
Introduction to Security

James Joshi

Assistant Professor, SIS

Lecture 5

September 27, 2007

Security Policies
Confidentiality Policies

Re-Cap

- Decidable vs Undecidable?
- Safety – leakage of rights
- HRU results:
  - Systems with mono-operational commands
    - k <= n*(?)(?) + 1?
  - Generic Safety problem
    - Turing machine ? Safety

Today's Objectives

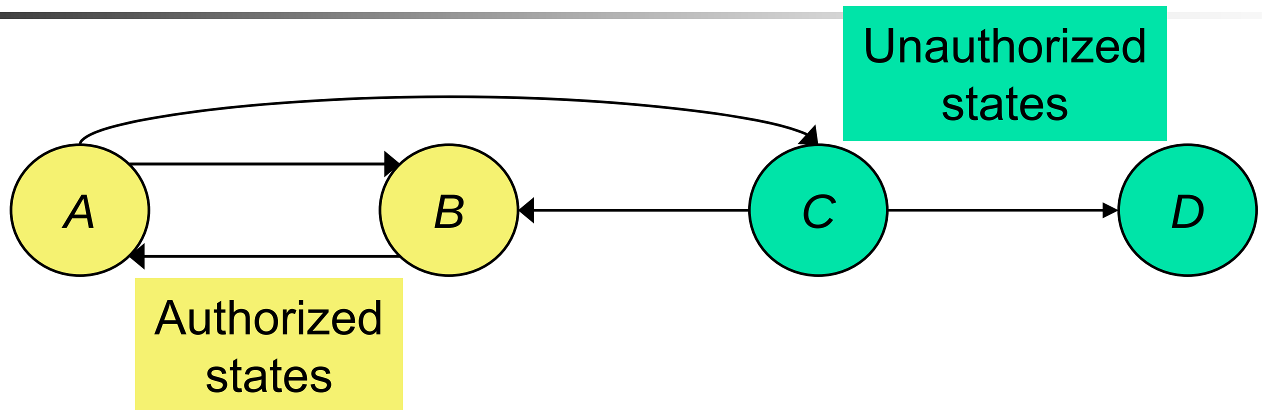
- Understanding/defining security policy and nature of trust
- Overview of different policy models
- Understand and analyze the lattice structure
- Define/Understand existing Bell-LaPadula model
  - how lattice helps?

Security Policies

Security Policy

- Defines what it means for a system to be secure
- Formally: Partitions a system into
  - Set of secure (authorized) states
  - Set of non-secure (unauthorized) states
- Secure system is one that
  - Starts in authorized state
  - Cannot enter unauthorized state

Secure System - Example



- Is this Finite State Machine Secure?
  - *A* is start state ?
  - *B* is start state ?
  - *C* is start state ?
  - How can this be made secure if not?
  - *Suppose A*, *B*, and *C* are authorized states ?

Additional Definitions:

- Security breach: system enters an unauthorized state
- Let *X* be a set of entities, *I* be information.
  - *I* has **confidentiality** with respect to *X* if no member of *X* can obtain information on *I*
  - *I* has **integrity** with respect to *X* if all members of *X* trust *I*
    - Trust *I*, its conveyance and storage (data integrity)
    - *I* maybe origin information or an identity (authentication)
    - *I* is a resource – its integrity implies it functions as it should (assurance)
  - *I* has **availability** with respect to *X* if all members of *X* can access *I*
    - Time limits (quality of service)

Confidentiality Policy

- Also known as *information flow*
  - Transfer of rights
  - Transfer of information without transfer of rights
  - Temporal context
- Model often depends on trust
  - Parts of system where information *could* flow
  - Trusted entity must participate to enable flow
- Highly developed in Military/Government

Integrity Policy

- Defines how information can be altered
  - Entities allowed to alter data
  - Conditions under which data can be altered
  - Limits to change of data
- Examples:
  - Purchase over $1000 requires signature
  - Check over $10,000 must be approved by one person and cashed by another
    - *Separation of duties :* for preventing fraud
- Highly developed in commercial world

Trust

- Theories and mechanisms rest on some trust assumptions
- 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

- Formal verification provides a 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?

Security Mechanism

- Policy describes what is allowed
- Mechanism
  - Is an entity/procedure that enforces (part of) policy
- Example Policy: Students should not copy homework
  - Mechanism: Disallow access to files owned by other users

Security Model

- A model that represents a particular policy or set of policies
  - Abstracts details relevant to analysis
  - Focus on specific characteristics of policies
    - E.g., Multilevel security focuses on information flow control

Security policies

- Military security policy
  - Focuses on confidentiality
- Commercial security policy
  - Primarily Integrity
  - Transaction-oriented
    - 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

Access Control

- Discretionary Access Control (DAC)
  - Owner determines access rights
  - Typically *identity-based access control*: Owner specifies other users who have access
- Mandatory Access Control (MAC)
  - Rules specify granting of access
  - Also called *rule-based access control*

Access Control

- Originator Controlled Access Control (ORCON)
  - Originator controls access
  - *Originator need not be owner!*
- Role Based Access Control (RBAC)
  - Identity governed by role user assumes

Lattice
Confidentiality Policies

Lattice

- Sets
  - Collection of unique elements
  - Let S, T be sets
    - Cartesian product: S x T = {(a, b) | a $\in$ A, b $\in$ B}
    - A set of order pairs
- Binary relation R from S to T is a subset of S x T
- Binary relation R on S is a subset of S x S

Lattice

- If (a, b) $\in R$ we write aRb
  - Example:
    - R is "less than equal to" ($\leq$)
    - For S = {1, 2, 3}
      - Example of R on S is {(1, 1), (1, 2), (1, 3), ????)
  - (1, 2) $\in R$ is another way of writing $1 \leq 2$

Lattice

- Properties of relations
  - Reflexive:
    - if $a\mathcal{R}a$ for all $a \in S$
  - Anti-symmetric:
    - if $a\mathcal{R}b$ and $b\mathcal{R}a$ implies a = b for all $a, b \in S$
  - Transitive:
    - if $a\mathcal{R}b$ and $b\mathcal{R}c$ imply that $a\mathcal{R}c$ for all $a, b, c \in S$

  - Exercise:
    - Which properties hold for “less than equal to” ($\leq$)?
    - Draw the Hasse diagram
      - Captures all the relations

Lattice

- Total ordering:
  - when the relation orders all elements
  - E.g., “less than equal to” ($\leq$) on natural numbers
- Partial ordering (poset):
  - the relation orders only some elements not all
  - E.g. “less than equal to” ($\leq$) on complex numbers; Consider (2 + 4i) and (3 + 2i)

Lattice

- Upper bound $(u, a, b \in S)$
  - u is an upper bound of a and b means aRu and bRu
  - Least upper bound : $lub(a, b)$ *closest upper bound*
- Lower bound $(l, a, b \in S)$
  - l is a lower bound of a and b means lRa and lRb
  - Greatest lower bound : $glb(a, b)$ *closest lower bound*

Lattice

- A lattice is the combination of a set of elements *S* and a relation *R* meeting the following criteria
  - R is reflexive, antisymmetric, and transitive on the elements of S
  - For every $s, t \in$ S, there exists a greatest lower bound
  - For every $s, t \in$ S, there exists a lowest upper bound
- Some examples
  - S = {1, 2, 3} and R = $\leq$?
  - S = {2+4i; 1+2i; 3+2i, 3+4i} and R = $\leq$?

Confidentiality Policy

- Also known as information flow policy
  - Integrity is secondary objective
  - Eg. Military mission “date”
- Bell-LaPadula Model
  - Formally models military requirements
    - Information has sensitivity levels or classification
    - Subjects have clearance
    - Subjects with clearance are allowed access
  - Multi-level access control or mandatory access control

Bell-LaPadula: Basics

- Mandatory access control
  - Entities are assigned security levels
  - Subject has security clearance $L(s) = l_s$
  - Object has security classification $L(o) = l_o$
  - Simplest case: Security levels are arranged in a linear order $l_i < l_{i+1}$
- Example

  Top secret > Secret > Confidential >Unclassified

"No Read Up"

- Information is allowed to flow *up*, not *down*
- *Simple security property:*
  - *s* can read *o* if and only if
    - $l_o \leq l_s$ and
    - *s* has discretionary read access to *o*
  - Combines mandatory *(security levels)* and discretionary *(permission required)*
  - Prevents subjects from reading objects at higher levels (*No Read Up rule*)

“No Write Down”

- Information is allowed to flow *up*, not *down*
- **property*
  - *s* can write *o* if and only if
    - $l_s \leq l_o$ and
    - *s* has write access to *o*
  - Combines mandatory *(security levels)* and discretionary *(permission required)*
  - Prevents subjects from writing to objects at lower levels (*No Write Down rule*)

Example

security level	*subject*	*object*
Top Secret	Tamara	Personnel Files
Secret	Samuel	E-Mail Files
Confidential	Claire	Activity Logs
Unclassified	Ulaley	Telephone Lists

- Tamara can *read* which objects? And *write*?
- Claire cannot read which objects? And *write*?
- Ulaley can *read* which objects? And *write*?

Access Rules

- Secure system:
  - One in which both the properties hold
- Theorem:
  - Let Σ be a system with secure initial state σ_0,
  - *T* be a set of state transformations
  - If every element of *T* follows rules, every state σ_i secure
  - Proof - induction

Categories

- Total order of classifications not flexible enough
  - Alice cleared for missiles; Bob cleared for warheads; Both cleared for targets
- Solution: Categories
  - Use set of compartments (from power set of compartments)
  - Enforce "*need to know*" principle
  - Security levels (security level, category set)
    - (Top Secret, {Nuc, Eur, Asi})
    - (Top Secret, {Nuc, Asi})

Lattice of categories

- Combining with clearance:
  - (L,C) *dominates* $(L',C') \Leftrightarrow L' \leq L$ and $C' \subseteq C$
  - Induces lattice of security levels
- Examples of levels
  - (Top Secret, {Nuc,Asi}) *dom* (Secret, {Nuc}) ?
  - (Secret, {Nuc, Eur}) *dom* (Topsecret, {Nuc,Eur}) ?
  - (Top Secret, {Nuc}) *dom* (Confidential, {Eur}) ?

Exercise: Hesse diagram for:
compartments: NUC, US, EU;

Exercise: Hesse diagram for:
Security levels: TS, S, C
Compartments US, EU;

Access Rules

- *Simple Security Condition*: *S* can read *O* if and only if
  - *S dominate O* and
  - *S* has read access to *O*
- **-Property*: *S* can write *O* if and only if
  - *O dom S* and
  - *S* has write access to *O*
- Secure system: One with above properties
- Theorem: Let Σ be a system with secure initial state σ_0, *T* be a set of state transformations
  - If every element of *T* follows rules, every state σ_i secure

Communication across level

- Communication is needed between
  - Subject at higher level and a subject at the lower levels
    - Need write down to a lower object
- One mechanism
  - Subjects have *max* and *current* levels
    - *max* must dominate *current*
  - Subjects decrease clearance level

Read & write

- Conventional use
  - “Read” – allowing information to flow from object being read to the subject reading
    - Read includes Execute
  - “Write” – allowing information to flow from the subject writing to the object being written
    - Write includes Append

- Could change based on the requirement and the model instantiated based on that.

Problem: No write-down

Cleared subject can't communicate to non-cleared subject

- Any write from l_i to l_k, $i > k$, would violate *-property
  - Subject at l_i can only write to l_i and above
- Any read from l_k to l_i, $i > k$, would violate simple security property
  - Subject at l_k can only read from l_k and below
- Subject at level l_i can't write something readable by subject at l_k
  - Not very practical

Principle of Tranquility

- Should we change classification levels?
- Raising object's security level
  - Information once available to some subjects is no longer available
  - Usually assumes information has already been accessed
  - Simple security property violated? Problem?

Principle of Tranquility

- Lowering object's security level
  - Simple security property violated?
  - The *declassification problem*
  - Essentially, a "write down" violating *-property
  - Solution: define set of trusted subjects that *sanitize* or remove sensitive information before security level is lowered

Types of Tranquility

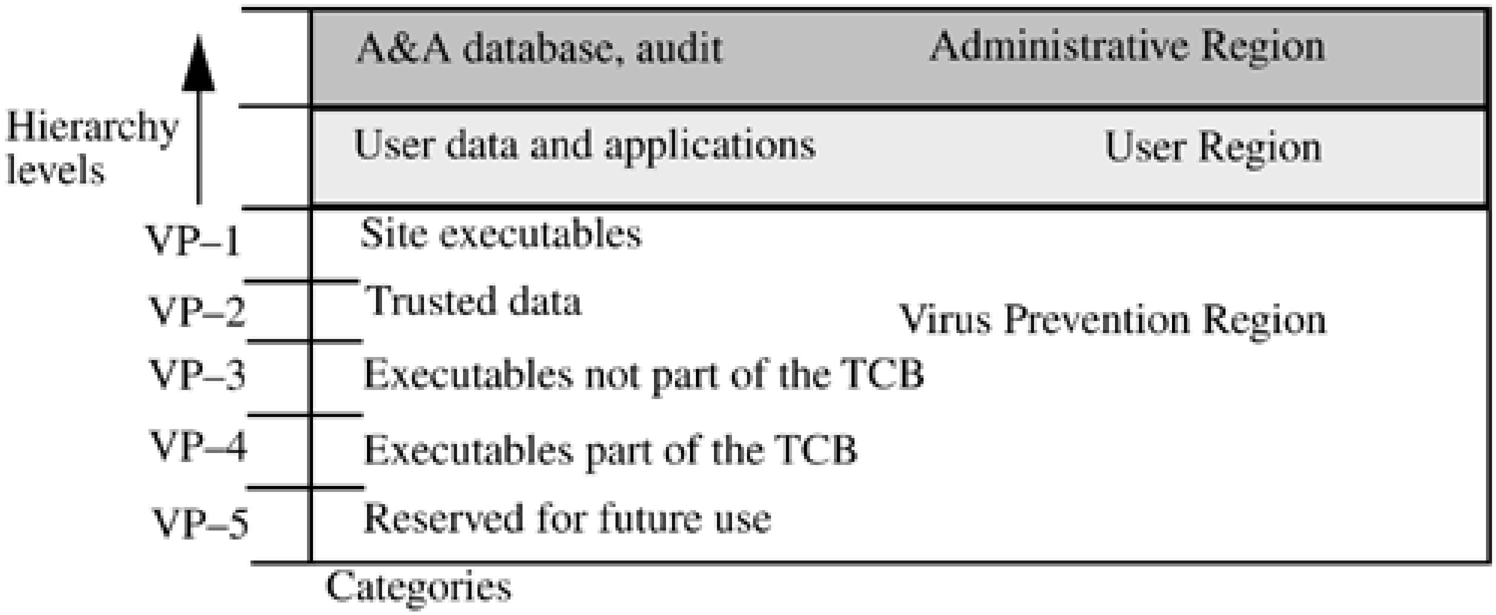
- Strong Tranquility
  - The clearances of subjects, and the classifications of objects, do not change during the lifetime of the system
- Weak Tranquility
  - The clearances of subjects, and the classifications of objects, do not change in a way that violates the simple security condition or the *-property during the lifetime of the system

Example

- DG/UX System
  - Only a trusted user (security administrator) can lower object's security level
  - In general, process MAC labels cannot change
    - If a user wants a new MAC label, needs to initiate new process
    - Cumbersome, so user can be designated as able to change process MAC label within a specified range

DG/UX Labels

- Lowest upper bound: IMPL_HI
- Greatest lower bound: IMPL_LO



DG/UX

- Once you login
  - MAC label that of user in Authorization and Authentication (A&A) Databases
- When a process begins
  - It gets its parent's MAC label
- Reading up and writing up not allowed

DG/UX

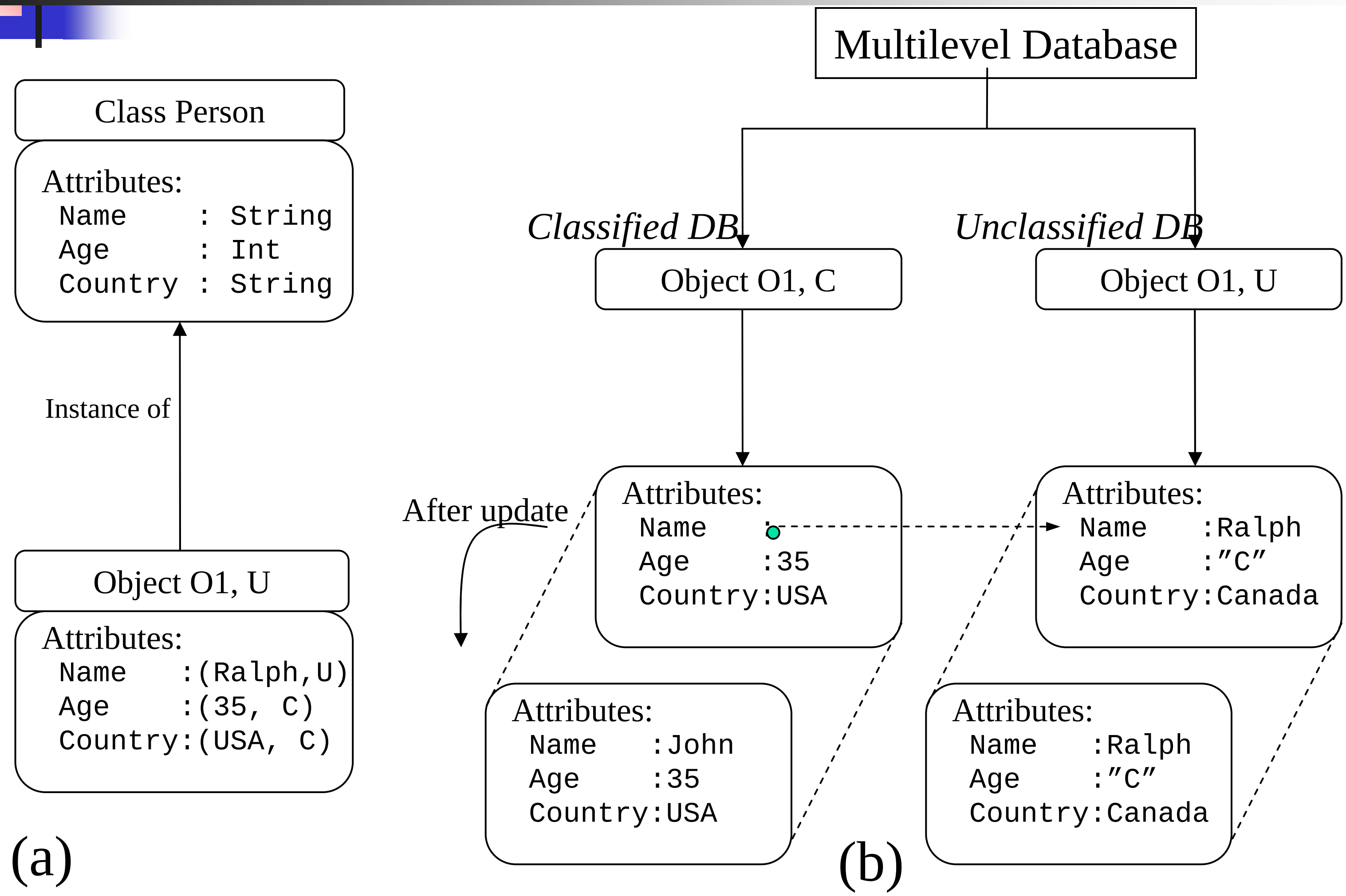
- S:MAC_A creates O
  - If O:MAC_B already exists
    - Fails if MAC_B dom MAC_A
- Creating files in a directory
  - Only programs with the same level as the directory can create files in the directory
  - Problems with /tmp and /var/mail
  - Solution: use multilevel directory:
    - a directory with a subdirectory for each level (hidden)
    - If process with MAC_A creates a file – put in subdirecotry with label MAC_A
    - Reference to parent directory of a file refers to the hidden directory

DG/UX

- Provides a range of MAC labels
  - Called MAC Tuples: [Lower, Upper]
    - [(S, {Europe}), (TS, {Europe})]
    - [(S, ∅), (TS, {Nuclear, Europe, Asia})]

    - Objects can have a tuple as well as a required MAC label
      - Tuple overrides
    - A process can read an object if its MAC label grants it read access to the upper bound
    - A process can read an object if its MAC label grants it write access to the lower bound

Multiview Model of MLS



Biba's Integrity Policy Model

- Based on Bell-LaPadula
  - Subject, Objects have
    - Integrity Levels with dominance relation
  - Higher levels
    - more reliable/trustworthy
    - More accurate

Biba's model

- Strict Integrity Policy (dual of Bell-LaPadula)
  - s can **read** $o \leftrightarrow i(s) \leq i(o)$ (no read-down)
  - s can **write** $o \leftrightarrow i(o) \leq i(s)$ (no write-up)
  - s_1 can **execute** $s_2 \leftrightarrow i(s_2) \leq i(s_1)$

Low-water-mark

- Low-Water-Mark Policy
  - s can **write** $o \leftrightarrow i(o) \le i(s)$
    - prevents writing to higher level
  - s **reads** $o \rightarrow i'(s) = min(i(s), i(o))$
    - drops subject's level
  - s_1 can **execute** $s_2 \leftrightarrow i(s_2) \le i(s_1)$
    - Why?

Summary

- Trust assumptions should be properly understood
- Lattice structure provides basis for representing information flow or confidentiality policies
  - Need to know
- Biba's integrity model is dual of BLP