Introduction to Computer Security

Lecture 4
Confidentiality and Integrity Policies

September 18, 2003

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
    - $l_o = l_s$ and
    - *s* has 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"

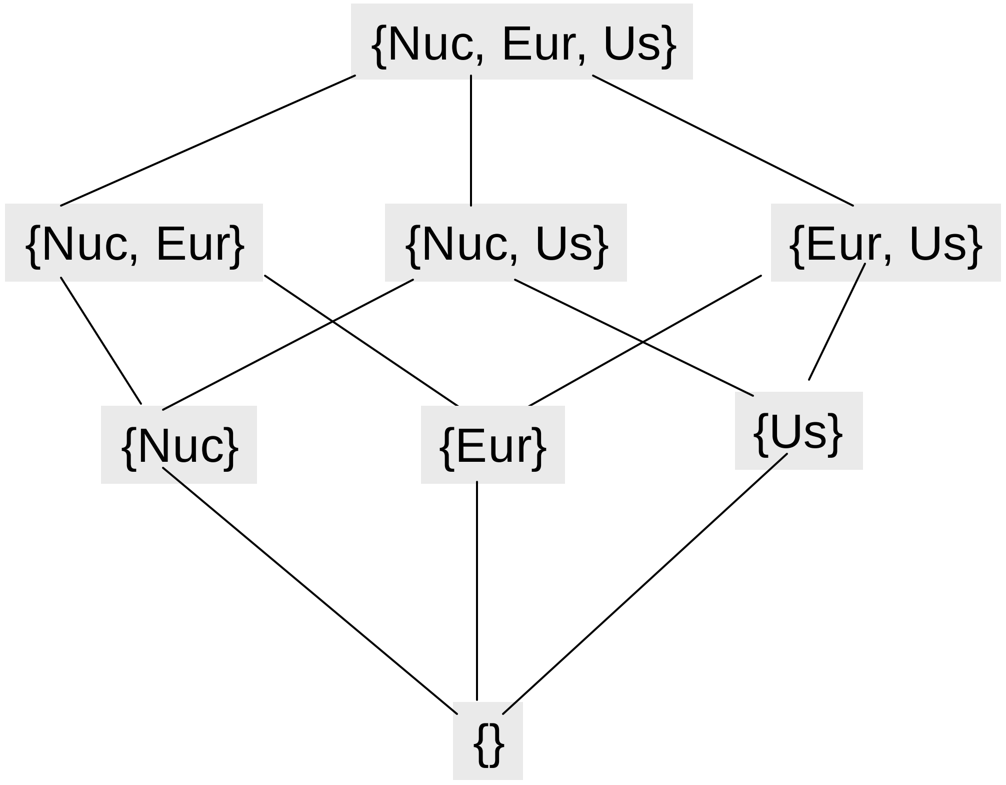
- Information is allowed to flow *up*, not *down*
- ***property**
  - *s* can write *o* if and only if
    - $l_s = l_o$ and
    - *s* has write access to *o*
  - Combines mandatory *(security levels)* and discretionary *(permission required)*
  - Prevents subjects from writing to objects at lower levels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

Bell LaPadula Model 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 (level, category set)
    - (Top Secret, {Nuc, Eur, Asi})
    - (Top Secret, {Nuc, Asi})
- Dominates relation
  - (L,C) *dominates* (L',C') $\Leftrightarrow L' = L$ and $C' \subseteq C$
  - Induces lattice of security levels

Lattice of categories

- Examples of levels
  - (Top Secret, {Nuc,Asi}) *dom* (Secret, {Nuc})?
  - (Secret, {Nuc, Eur}) *dom* (Confidential, {Nuc,Eur})?
  - (Top Secret, {Nuc}) *dom* (Confidential, {Eur}) ?
- Bounds
  - Greatest lower, *glb*
  - Lowest upper, *lub*
  - *glb* of {Nuc, Us} & {Eur, Us}?
  - *lub* of {Nuc, Us} & {Eur, Us}?



Access Rules

- *Simple Security Condition*: *S* can read *O* if and only if
  - *Clearance of S dominates classification of O* and
  - *S* has read access to *O*
- **-Property*: *S* can write *O* if and only if
  - *Classification of O dominates clearance of S* and
  - *S* has write access to *O*
- Secure system: One with above properties
- Theorem: Let S be a system with secure initial state s_0, *T* be a set of state transformations
  - If every element of *T* follows rules, every state s_i secure

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 i can't write something readable by subject at k
  - Not very practical
  - Solution: Allow *maximum level* and *current level*.

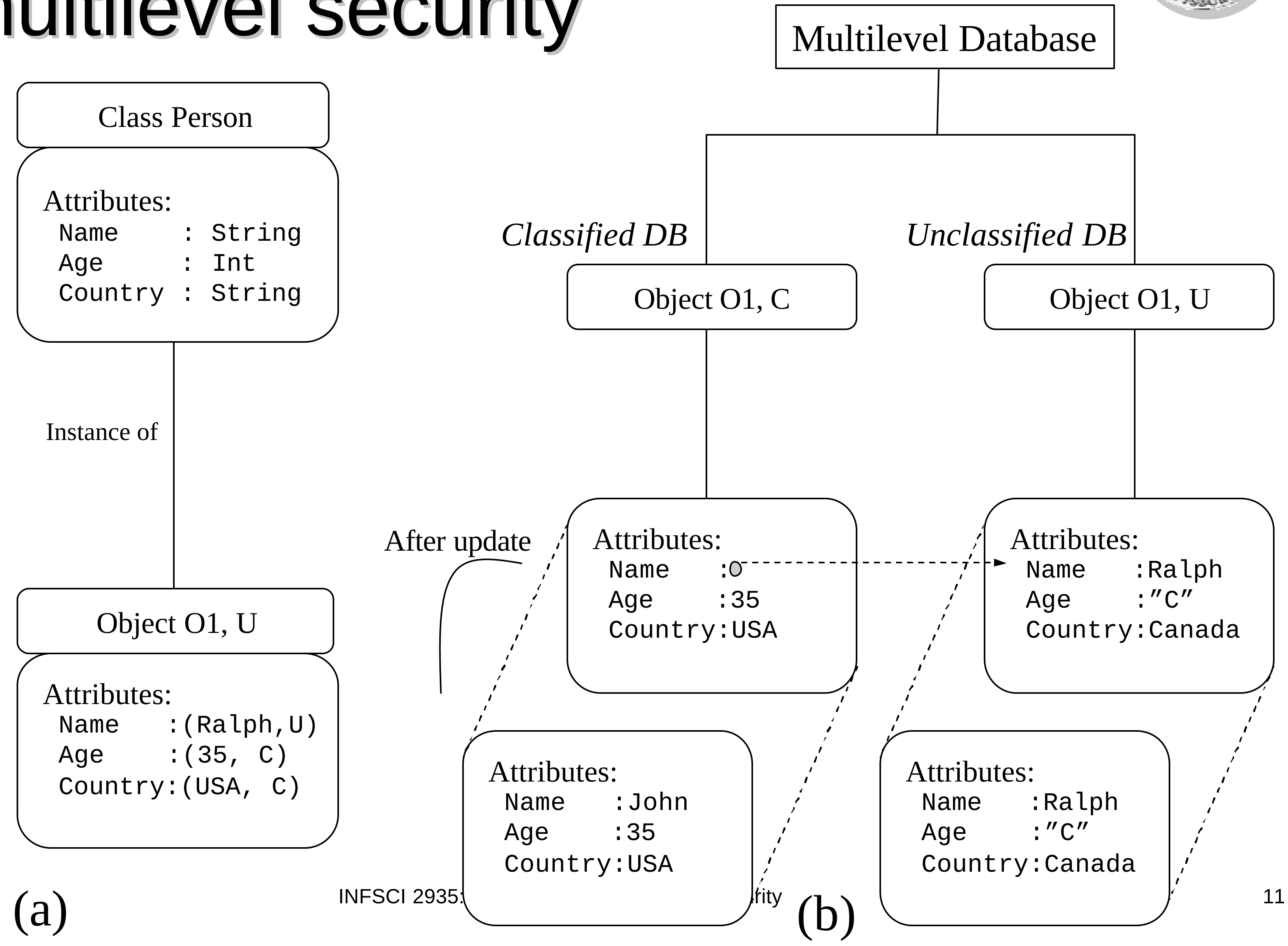
Principle of Tranquility

- 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?
- 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

Multiview Model of multilevel security



Integrity Policies

Overview

- Requirements
  - Very different than confidentiality policies
- Biba’s models
  - Low-Water-Mark policy
  - Ring policy
  - Strict Integrity policy
- Lipner’s model
  - Combines Bell-LaPadula, Biba
- Clark-Wilson model

Requirements of Commercial Integrity Policies (Lipner)

1. Users will not write their own programs, but will use existing production programs and databases.
2. Programmers will develop and test programs on a nonproduction system; if they need access to actual data, they will be given production data via a special process, but will use it on their development system.
3. A special process must be followed to install a program from the development system onto the production system.
4. The special process in requirement 3 must be controlled and audited.
5. The managers and auditors must have access to both the system state and the system logs that are generated.

Integrity Policy: Principles of operation

- Requirements induce principles of operation:
  - Separation of Duty: Single person should not be allowed to carry out all steps of a critical function
    - Moving a program from Dev. to Prod. system
    - Developer and Certifier (installer) of a program
    - Authorizing checks and cashing it
  - Separation of function
    - Do not process production data on development system
  - Auditing
    - Emphasis on recovery and accountability
    - Controlled/audited process for updating code on production system

Biba's Integrity Policy Model

- Based on Bell-LaPadula
  - Subject, Objects
  - Integrity Levels with dominance relation
    - Higher levels
      - more reliable/trustworthy
      - More accurate
- Information transfer path:
  Sequence of subjects, objects where
  - s_i **r** o_i
  - s_i **w** o_{i+1}

Policies

- Low-Water-Mark Policy
  - $s \, \mathbf{w} \, o \Leftrightarrow i(o) = i(s)$ prevents writing to higher level
  - $s \, \mathbf{r} \, o \Rightarrow i'(s) = min(i(s), i(o))$ drops subject's level
  - $s_1 \, \mathbf{x} \, s_2 \Leftrightarrow i(s_2) = i(s_1)$ prevents executing higher level objects
- Ring Policy
  - $s \, \mathbf{r} \, o$ allows any subject to read any object
  - $s \, \mathbf{w} \, o \Leftrightarrow i(o) = i(s)$ (same as above)
  - $s_1 \, \mathbf{x} \, s_2 \Leftrightarrow i(s_2) = i(s_1)$
- Biba's Model: Strict Integrity Policy (dual of Bell-LaPadula)
  - $s \, \mathbf{r} \, o \Leftrightarrow i(s) = i(o)$ (no read-down)
  - $s \, \mathbf{w} \, o \Leftrightarrow i(o) = i(s)$ (no write-up)
  - $s_1 \, \mathbf{x} \, s_2 \Leftrightarrow i(s_2) = i(s_1)$
- Theorem for each:
  - If there is an information transfer path from object o_1 to object o_{n+1}, then the enforcement of the policy requires that $i(o_{n+1}) = i(o_1)$ for all n>1

LOCUS and Biba

- Goal:
  - prevent untrusted software from altering data or other software (limit execution domain)
- Approach: make levels of trust explicit
  - *credibility rating* based on estimate of software's trustworthiness (0 untrusted, *n* highly trusted)
  - *trusted file systems* contain software with a single credibility level
  - User/process has *risk level* or highest credibility level at which process can execute
  - Must use *run-untrusted* command to run software at lower credibility level

Lipner: Integrity Matrix

- BLP + Biba to conform to commercial requirement
- Security Levels
  - Audit: AM
    - Audit/management functions
  - System Low: SL
    - Everything else; any process can read information at this level
- Categories
  - **D**evelopment (not yet in production use)
  - **P**roduction **C**ode (production processes and programs)
  - **P**roduction **D**ata (data covered by the integrity policy)
  - **S**ystem **D**evelopment (system programs under development)
  - Software **T**ools (programs in production system not related to sensitive/protected data)
- Follow Bell-LaPadula security properties

Lipner: Integrity Matrix

Users:	Clearance
Ordinary	(SL,{PC, PD})
Developers	(SL,{D,T})
System Programmers	(SL,{SD, T})
System Managers/Aud.	(AM,{D,PC,PD,SD,T})
Controllers	(SL,{D,PC,PD,SD,T} + downgrade prv

Objects	Classification
Development code/data	(SL,{D,T})
Production code	(SL,{PC})
Production data	(SL,{PC,PD})
Tools	(SL,{T})
System Programs	(SL,∅)
System Program update	(SL,{SD,T})
Logs	(AM, {…})

Check against the requirement

- Users will not write their own programs, but will use existing production programs and databases.
  - Users have no access to T, so cannot write their own programs
- Programmers will develop and test programs on a non production system; if they need access to actual data, they will be given production data via a special process, but will use it on their development system.
  - Applications programmers have no access to PD, so cannot access production data; if needed, it must be put into D (downgrade), requiring the system controller to intervene

Check against the requirement

- A special process must be followed to install a program from the development system onto the production system.
  - Installing a program requires downgrade procedure (from D to PC), so only system controllers can do it
- The special process in requirement 3 must be controlled and audited.
  - Control: only system controllers can downgrade; audit: any such downgrading must be logged
- The managers and auditors must have access to both the system state and the system logs that are generated.
  - System management and audit users are in AM and so have access to system state and logs

Problem

- Too inflexible
  - System managers cannot run programs for repairing inconsistent or erroneous production database
  - A program for repairing an inconsistent database cannot be application level software
    - An integrity issue
- So add more …

Lipner’s full model
Introduce integrity levels

- Integrity classifications (highest to lowest)
  - ISP (System Program): for system programs
  - IO (Operational): production programs, development software
  - ISL (System Low): users get this on log in
- Integrity categories (distinguish between development and production)
  - ID (Development): development entities
  - IP (Production): production entities

Simplify Bell-LaPadula

- Reduce security categories to 3:
  - SP (Production): production code, data
  - SD (Development): same as D
  - SSD (System Development): same as old SD
    - Remove T
      - Earlier category T allowed application developers and system programmers to use the same programs without being able to alter those programs.
      - The new integrity categories distinguish between development and production, so they serve the purpose of software tools category
    - Collapse PC and PD into SP category

Users and Levels

Subjects	**Security Level (same as before)**	**Integrity Level**
Ordinary users	(SL, { SP })	(ISL, { IP })
Application developers	(SL, { SD })	(ISL, { ID })
System programmers	(SL, { SSD })	(ISL, { ID })
System managers and auditors	(AM, { SP, SD, SSD })	(ISL, { IP, ID})
System controllers	(SL, { SP, SD }) and downgrade privilege	(ISP, { IP, ID})
Repair	(SL, { SP })	(ISL, { IP })

Key Ideas for Assigning Integrity Levels to Objects

- Security clearances of subjects same as without integrity levels
- Ordinary users need to modify production data, so ordinary users must have write access to integrity category IP
- Ordinary users must be able to write production data but not production code; integrity classes allow this

Objects and Classifications

Objects	**Security Level (earlier category)**	**Integrity Level**
Development code/test data	(SL, { SD }) (D, T)	(ISL, { IP})
Production code	(SL, { SP }) (PC)	(IO, { IP }) ?
Production data	(SL, { SP }) (PC, PD)	(ISL, { IP }) ?
Software tools	(SL, ∅) (T)	(IO, { ID })
System programs	(SL, ∅) ∅	(ISP, { IP, ID })
System programs in modification	(SL, { SSD }) (SD, T)	(ISL, { ID })
System and application logs	(AM, { *appropriate* })	(ISL, ∅)
Repair	(SL, {SP})	(ISL, { IP })

What can an ordinary user do?

- Ordinary users can : (SL, { SP }) (ISL, { IP })
  - Read and write production data (same security integrity levels)
  - Read production code
    - same classification &
    - (IO, IP) *dom* (ISL, {IP})
  - System program
    - (SL, {SP}) *dom* (SL, $\emptyset$) &
    - (ISP, {IP,ID}) *dom* {ISL, {IP})
  - Repair objects (same levels)
  - Write (not read) the system and application log
    - (AM, {SP}) *dom* (SL, {SP}) &
    - (ISL, {IP}) *dom* {ISL, $\emptyset$})

Clark-Wilson Integrity Model

- Transactions as the basic operation
- Integrity defined by a set of constraints
  - Data in a *consistent* or valid state when it satisfies these
- Example: Bank
  - *D* today's deposits, *W* withdrawals, *YB* yesterday's balance, *TB* today's balance
  - Integrity constraint: $D + YB - W$
- *Well-formed transaction*
  - A series of operations that move system from one consistent state to another
  - State before transaction consistent $\Rightarrow$ state after transaction consistent
- Issue: who examines, certifies transactions done correctly?
  - Separation of duty is crucial

Clark/Wilson Model Entities

- **C**onstrained **D**ata **I**tems (CDI) : data subject to Integrity Control
  - Eg. Account balances
  - *Integrity constraints* constrain the values of the CDIs
- **U**nconstrained **D**ata **I**tems (UDI): data not subject to IC
  - Eg. Gifts given to the account holders
- **I**ntegrity **V**erification **P**rocedures (IVP)
  - Test CDIs' conformance to integrity constraints at the time IVPs are run (checking that accounts balance)
- **T**ransformation **P**rocedures (TP); E.g.,
  - Depositing money
  - Withdrawing money
  - Money transfer etc.

Clark/Wilson: Certification/Enforcement Rules

- C1: When any IVP is run, it must ensure all CDIs are in valid state
- C2: A TP must transform a set of CDIs from a valid state to another valid state
  - TR must not be used on CDIs it is not certified for
- E1: System must maintain certified relations
  - TP/CDI sets enforced
- E2: System must control users
  - *user*/TP/CDI mappings enforced

Clark/Wilson: Certification/Enforcement Rules

- C3: Relations between (*user*, TP, {CDI}) must support separation of duty
- E3: Users must be authenticated to execute TP
  - Note, unauthenticated users may manipulate UDIs
- C4: All TPs must log undo information to append-only CDI (to reconstruct an operation)
- C5: A TP taking a UDI as input must either reject it or transform it to a CDI
- E4: Only certifier of a TP may change the list of entities associated with that TP
  - Enforces separation of duty: if a user could create a TP and associate some set of entities and himself with that TP, he could have the TP perform some unauthorized act

Requirements of Commercial Integrity Policies (Lipner)

1. Users will not write their own programs, but will use existing production programs and databases.
2. Programmers will develop and test programs on a nonproduction system; if they need access to actual data, they will be given production data via a special process, but will use it on their development system.
3. A special process must be followed to install a program from the development system onto the production system.
4. The special process in requirement 3 must be controlled and audited.
5. The managers and auditors must have access to both the system state and the system logs that are generated.

Comparison With Requirements

1. Users can’t (only trusted personnel can) certify TPs, so CR5 and ER4 enforce this
2. Procedural, so model doesn’t directly cover it; but special process corresponds to using TP
   - No technical controls can prevent programmer from developing program on production system; usual control is to delete software tools
3. TP does the installation, trusted personnel do certification

Comparison With Requirements

4. CR4 provides logging; ER3 authenticates trusted personnel doing installation; CR5, ER4 control installation procedure
   - New program is UDI before certification, CDI (and TP) after
5. Log is CDI, so appropriate TP can provide managers, auditors access
   - Access to state handled similarly

Summary

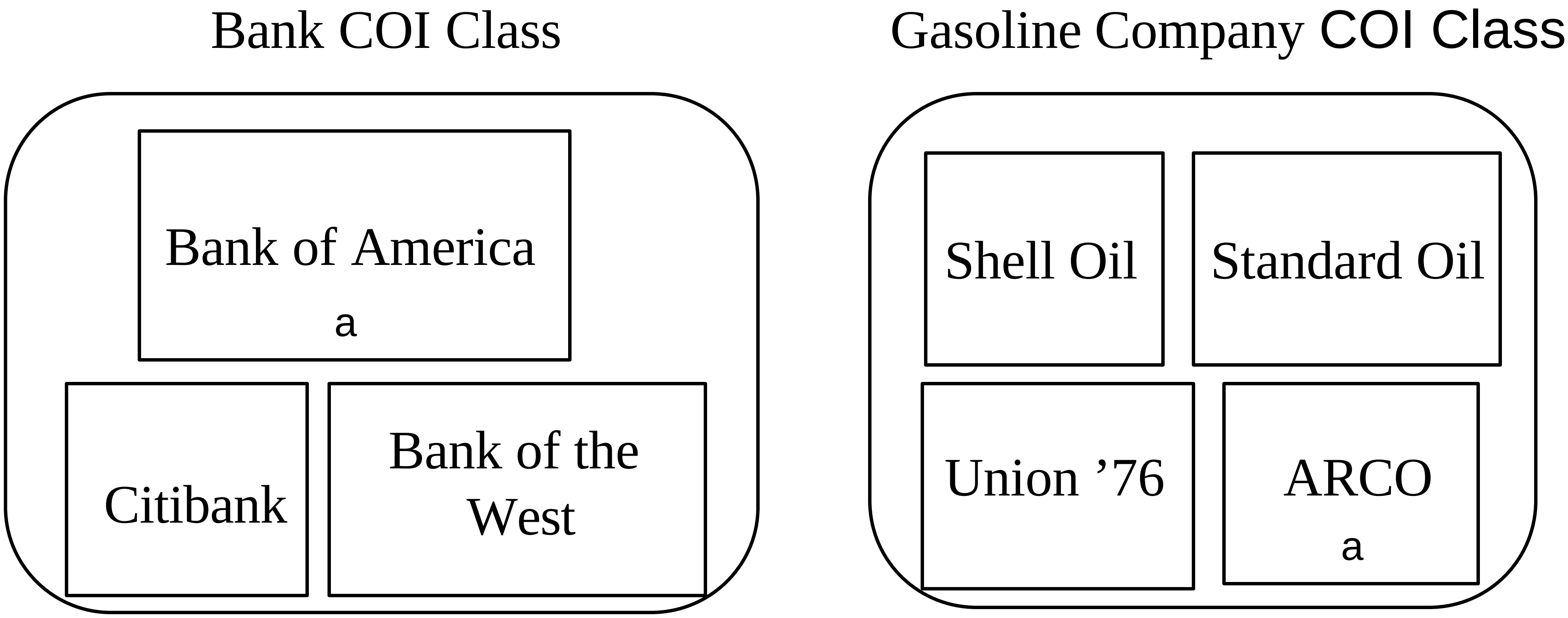
- Integrity policies deal with trust
  - As trust is hard to quantify, these policies are hard to evaluate completely
  - Look for assumptions and trusted users to find possible weak points in their implementation
- Biba, Lipner based on multilevel integrity
- Clark-Wilson introduce new ideas
  - Commercial firms do not classify data using multilevel scheme and they enforce separation of duty
  - Notion of certification is different from enforcement;
    - enforcement rules can be enforced,
    - certification rules need outside intervention, and
    - process of certification is complex and error prone

Hybrid Policies

Chinese Wall Model

- Supports confidentiality and integrity
  - Information can’t flow between items in a Conflict of Interest set
  - Applicable to environment of stock exchange or investment house
- Models conflict of interest
  - *Objects*: items of information related to a company
  - *Company dataset* (CD): contains objects related to a single company
    - Written *CD*(*O*)
  - *Conflict of interest class* (COI): contains datasets of companies in competition
    - Written *COI*(*O*)
    - Assume: each object belongs to exactly one *COI* class

Example


Bank COI Class
Gasoline Company COI Class
Bank of America
a
Citibank
Bank of the West
Shell Oil
Standard Oil
Union '76
ARCO
a

CW-Simple Security Property (Read rule)

- CW-Simple Security Property
  - *s* can read *o* $\Leftrightarrow$ one of the following holds
    - $\exists\, o' \in PR(s)$ such that $CD(o') = CD(o)$
    - $\forall\, o', o' \in PR(s) \Rightarrow COI(o') \neq COI(o)$, or
    - *o* has been “sanitized”

    ($o' \in PR(s)$ indicates *o’* has been previously read by s)
  - Public information may belong to a CD
    - As is publicly available, no conflicts of interest arise
    - So, should not affect ability of analysts to read
    - Typically, all sensitive data removed from such information before it is released publicly (called *sanitization*)

Writing

- Anthony, Susan work in same trading house
- Anthony can read BankOfAmercia's CD,
- Susan can read Bank CitiBanks's CD,
- Both can read ARCO's CD
- If Anthony could write to Gas' CD, Susan can read it
  - Hence, indirectly, she can read information from BankOfAmercia's CD, a clear conflict of interest

CW-*-Property (Write rule)

- CW-*- Property
  - *s* can read *o* ⇔ the following holds
    - The CW-simple security condition permits S to read O.
    - For all unsanitized objects o', s can read o' ⇒ *CD*(*o'*) = *CD*(*o*)

      Says that s can write to an object if all the (unsanitized) objects it can read are in the same dataset
  - Anthony can read both CDs hence condition 1 is met
  - He can read unsanitized objects of BankOfAmercia, hence condition 2 is false
    - Hence Anthony can't write to objects in ARCO's CD.

Compare to Bell-LaPadula

- Fundamentally different
  - CW has no security labels, B-LP does
  - CW has notion of past accesses, B-LP does not
- Bell-LaPadula can capture state at any time
  - Each (COI, CD) pair gets security category
  - Two clearances, *S* (sanitized) and *U* (unsanitized) such that (*S dom U)*
  - Subjects assigned clearance for compartments without multiple categories corresponding to CDs in same COI class

    eg. If Susan can read the BankOfAmerica and ARCO CDs, her process would get clearance for compartment (U, {a, n})

Compare to Bell-LaPadula

- Bell-LaPadula cannot track changes over time
  - Susan becomes ill, Anna needs to take over
    - C-W history lets Anna know if she can
    - No way for Bell-LaPadula to capture this
- Access constraints change over time
  - Initially, subjects in C-W can read any object
  - Bell-LaPadula constrains set of objects that a subject can access
    - Can't clear all subjects for all categories, because this violates CW-simple security condition

Compare to Clark-Wilson

- Clark-Wilson Model covers integrity, CW consider only access control aspects
- If “subjects” and “processes” are interchangeable, a single person could use multiple processes to violate CW-simple security condition
- If “subject” is a specific person and includes all processes the subject executes, then consistent with Clark-Wilson Model

Clinical Information Systems Security Policy (Anderson)

- Intended for medical records
  - Conflict of interest not critical problem
  - Patient confidentiality, authentication of records and annotators, and integrity are
- Entities:
  - Patient: subject of medical records (or agent)
  - Personal health information: data about patient’s health or treatment enabling identification of patient
  - Clinician: health-care professional with access to personal health information while doing job

Assumptions and Principles

- Assumes health information involves 1 person at a time
  - Not always true; OB/GYN involves father as well as mother
- Principles derived from medical ethics of various societies, and from practicing clinicians

Access

- Principle 1: Each medical record has an access control list naming the individuals or groups who may read and append information to the record. The system must restrict access to those identified on the access control list.
  - Idea is that clinicians need access, but no-one else. Auditors get access to copies, so they cannot alter records

Access

- Principle 2: One of the clinicians on the access control list must have the right to add other clinicians to the access control list.
  - Called the *responsible clinician*

Access

- Principle 3: The responsible clinician must notify the patient of the names on the access control list whenever the patient’s medical record is opened. Except for situations given in statutes, or in cases of emergency, the responsible clinician must obtain the patient’s consent.
  - Patient must consent to all treatment, and must know of violations of security

Access

- Principle 4: The name of the clinician, the date, and the time of the access of a medical record must be recorded. Similar information must be kept for deletions.
    - This is for auditing. Don’t delete information; update it (last part is for deletion of records after death, for example, or deletion of information when required by statute). Record information about all accesses.

Creation

- Principle: A clinician may open a record, with the clinician and the patient on the access control list. If the record is opened as a result of a referral, the referring clinician may also be on the access control list.
  - Creating clinician needs access, and patient should get it. If created from a referral, referring clinician needs access to get results of referral.

Deletion

- Principle: Clinical information cannot be deleted from a medical record until the appropriate time has passed.
  - This varies with circumstances.

Confinement

- Principle: Information from one medical record may be appended to a different medical record if and only if the access control list of the second record is a subset of the access control list of the first.
  - This keeps information from leaking to unauthorized users. All users have to be on the access control list.

Aggregation

- Principle: Measures for preventing the aggregation of patient data must be effective. In particular, a patient must be notified if anyone is to be added to the access control list for the patient’s record and if that person has access to a large number of medical records.
  - Fear here is that a corrupt investigator may obtain access to a large number of records, correlate them, and discover private information about individuals which can then be used for nefarious purposes (such as blackmail)

Enforcement

- Principle: Any computer system that handles medical records must have a subsystem that enforces the preceding principles. The effectiveness of this enforcement must be subject to evaluation by independent auditors.
  - This policy has to be enforced, and the enforcement mechanisms must be auditable (and audited)

Compare to Bell-LaPadula

- Confinement Principle imposes lattice structure on entities in model
  - Similar to Bell-LaPadula
- CISS focuses on objects being accessed; B-LP on the subjects accessing the objects
  - May matter when looking for insiders in the medical environment

Compare to Clark-Wilson

- CDIs are medical records
- TPs are functions updating records, access control lists
- IVPs certify:
  - A person identified as a clinician is a clinician;
  - A clinician validates, or has validated, information in the medical record;
  - When someone is to be notified of an event, such notification occurs; and
  - When someone must give consent, the operation cannot proceed until the consent is obtained
- Auditing (CR4) requirement: make all records append-only, notify patient when access control list changed

ORCON

- Problem: organization creating document wants to control its dissemination
  - Example: Secretary of Defense writes a memo for distribution to her immediate subordinates, and she must give permission for it to be disseminated further. This is “originator controlled” (here, the “originator” is a person).

Requirements

- Subject $s \in S$ marks object $o \in O$ as ORCON on behalf of organization *X*. *X* allows *o* to be disclosed to subjects acting on behalf of organization *Y* with the following restrictions:
  1. *o* cannot be released to subjects acting on behalf of other organizations without *X*'s permission; and
  2. Any copies of *o* must have the same restrictions placed on it.

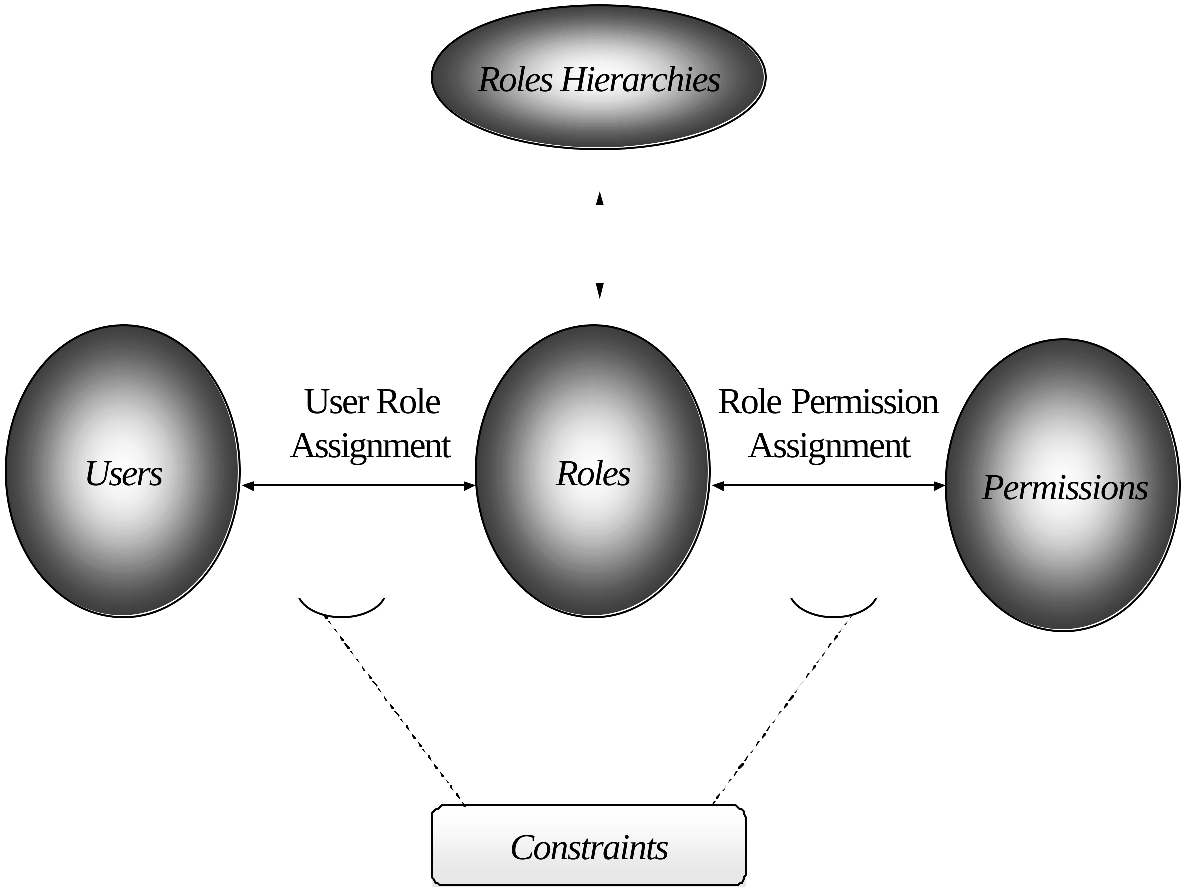
- DAC fails
  - Owner can set any desired permissions
    - This makes 2 unenforceable

MAC Fails

- First problem: category explosion
  - Category *C* contains *o*, *X*, *Y*, and nothing else.
  - If a subject $y \in Y$ wants to read *o*, $x \in X$ makes a copy *o*´.
  - Note *o*´ has category *C*. If *y* wants to give $z \in Z$ a copy, *z* must be in *Y*—by definition, it's not.
  - If *x* wants to let $w \in W$ see the document, need a new category *C*´ containing *o*, *X*, *W*.
- Second problem: abstraction
  - MAC classification, categories centrally controlled, and access controlled by a centralized policy
  - ORCON controlled locally

Role Based Access Control (RBAC)

- Access control in organizations is based on “roles that individual users take on as part of the organization”
- A role is “is a collection of permissions”



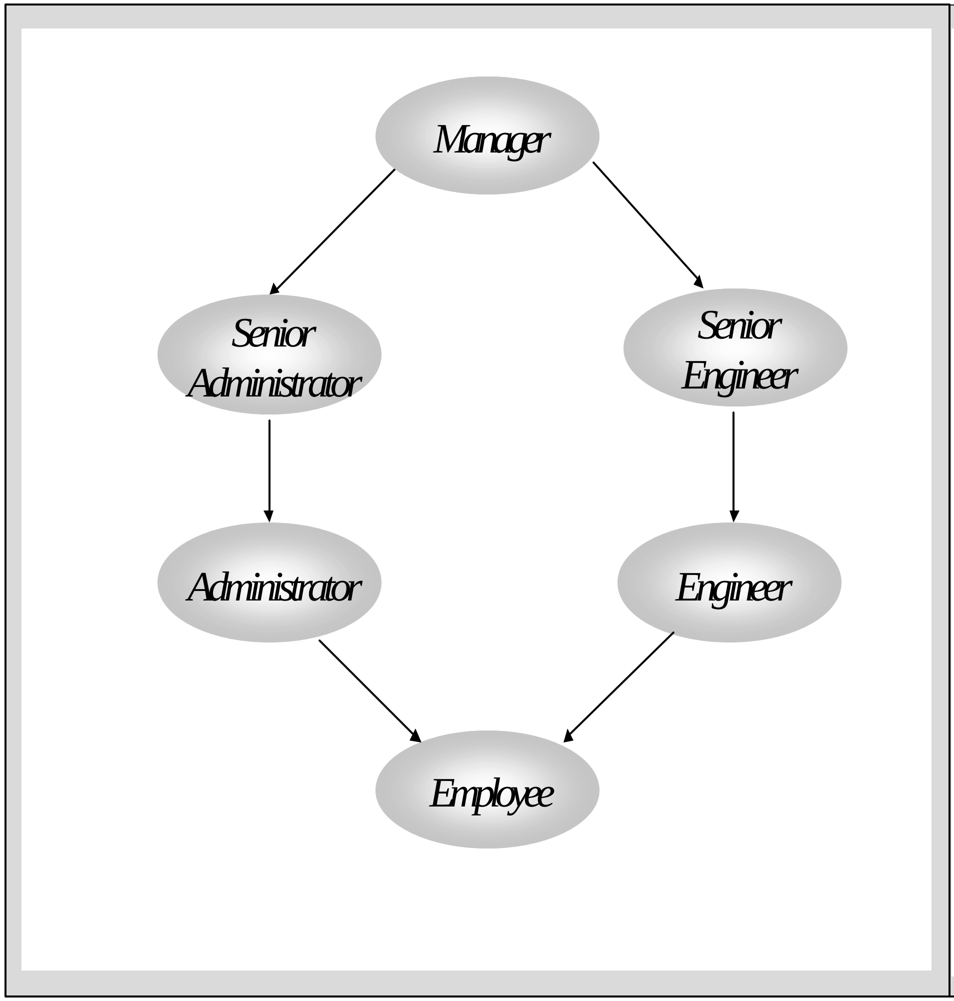
RBAC

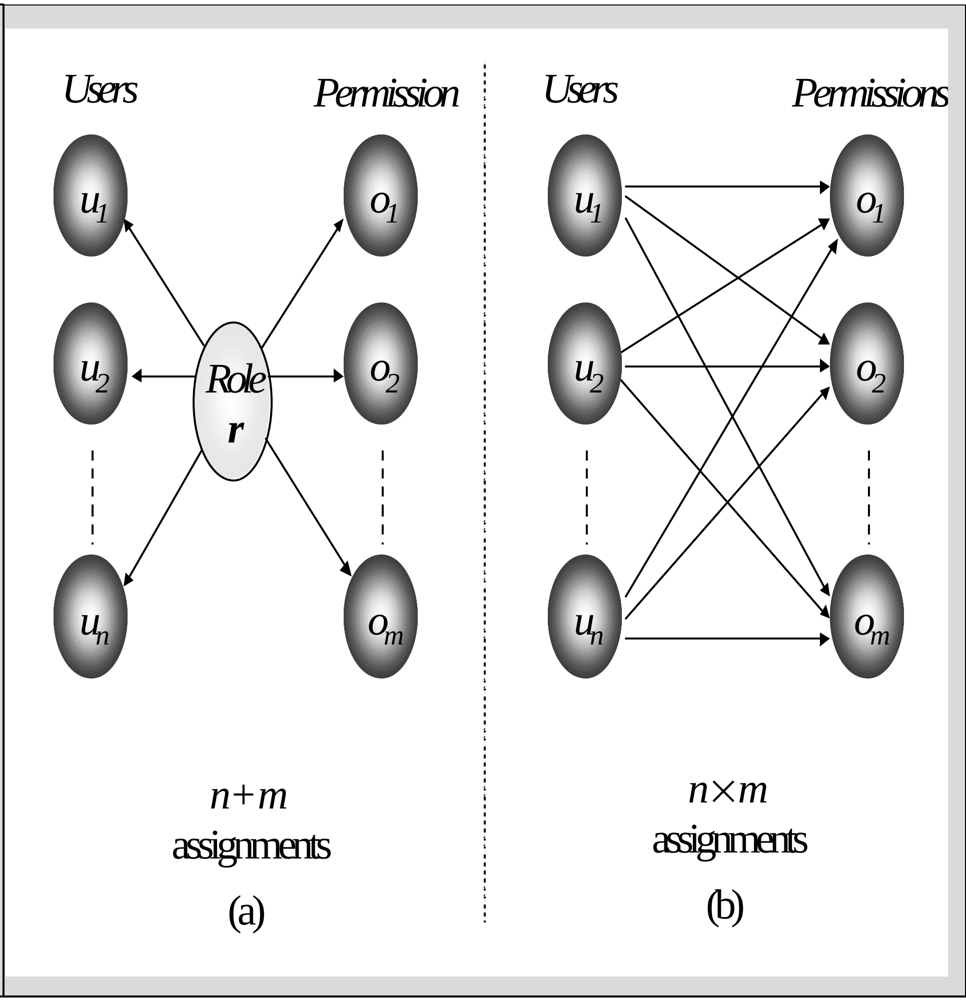
- Access depends on function, not identity
  - Example: Allison is bookkeeper for Math Dept. She has access to financial records. If she leaves and Betty is hired as the new bookkeeper, Betty now has access to those records. The role of “bookkeeper” dictates access, not the identity of the individual.

Advantages of RBAC

- Allows Efficient Security Management
  - Administrative roles, Role hierarchy
- Principle of least privilege allows minimizing damage
- Separation of Duties constraints to prevent fraud
- Allows grouping of objects
- Policy-neutral - Provides generality
- Encompasses DAC and MAC policies

RBAC


Manager
Senior Administrator
Senior Engineer
Administrator
Engineer
Employee


Users
Permission
u_1
u_2
u_n
Role $\boldsymbol{r}$
o_1
o_2
o_m
$n+m$ assignments
(a)
Users
Permissions
u_1
u_2
u_n
o_1
o_2
o_m
$n \times m$ assignments
(b)