CSE509: (Intro to) Systems Security

Fall 2012

Radu Sion

Integrity Policies
Hybrid Policies

Integrity Policies: Overview

- Requirements
  - Very different than confidentiality policies
- Biba's model
- Clark-Wilson model

Commercial Frameworks: Requirements

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 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.
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.

Biba Integrity Model

- Set of subjects S, objects O, integrity levels I, relation $\leq \subseteq I \times I$ holding when second dominates first
- min: $I \times I \rightarrow I$ returns lesser of integrity levels
- i: $S \cup O \rightarrow I$ gives integrity level of entity
- $\underline{\text{r}}$: $S \times O$ means $s \in S$ can read $o \in O$
- $\underline{\text{w}}$, $\underline{\text{x}}$ defined similarly

Intuition for Integrity Levels

- The higher the level, the more confidence
  - That a program will execute correctly
  - That data is accurate and/or reliable
- Note relationship between integrity and trustworthiness
- Important point: *integrity levels are **not** security levels*

Information Transfer Path

- An *information transfer path* is a sequence of objects $o_1, \ldots, o_{n+1}$ and corresponding sequence of subjects $s_1, \ldots, s_n$ such that $s_i \; \underline{\mathrm{r}} \; o_i$ and $s_i \; \underline{\mathrm{w}} \; o_{i+1}$ for all i, $1 \leq i \leq n$.
- Idea: information can "flow" from o_1 to o_{n+1} along this path by successive reads and writes

Low-Water-Mark Policy

- Idea: when s reads o, $i(s) = min(i(s), i(o))$; s can only write objects at lower levels
- Rules
  1. $s \in S$ can write to $o \in O$ if and only if $i(o) \leq i(s)$.
  2. If $s \in S$ reads $o \in O$, then $i'(s) = min(i(s), i(o))$, where $i'(s)$ is the subject's integrity level after the read.
  3. $s_1 \in S$ can execute $s_2 \in S$ if and only if $i(s_2) \leq i(s_1)$.

Information Flow and Model

If there is an information transfer path from $o_1 \in O$ to $o_{n+1} \in O$, enforcement of low-water-mark policy requires $i(o_{n+1}) \leq i(o_\mathrm{n})$ for all $n > 1$.

- proof: by induction

Problems

- Subjects' integrity levels decrease as system runs
  - Soon no subject will be able to access objects at high integrity levels
- Alternative: change object levels rather than subject levels
  - Soon all objects will be at the lowest integrity level
- Crux of problem: model prevents indirect modification
  - Because subject levels lowered when subject reads from low-integrity object

Ring Policy

- Idea: subject integrity levels static
- Rules
  1. $s \in S$ can write to $o \in O$ if and only if $i(o) \leq i(s)$.
  2. Any subject can read any object.
  3. $s_1 \in S$ can execute $s_2 \in S$ if and only if $i(s_2) \leq i(s_1)$.
- Eliminates indirect modification problem
- Same information flow result holds

Strict Integrity Policy ("Biba Model")

- Similar to Bell-LaPadula model
  1. $s \in S$ can read $o \in O$ iff $i(s) \le i(o)$
  2. $s \in S$ can write to $o \in O$ iff $i(o) \le i(s)$
  3. $s_1 \in S$ can execute $s_2 \in S$ iff $i(s_2) \le i(s_1)$
- Need to add compartments (and discretionary controls) to get full dual of Bell-LaPadula model
- Information flow result holds
- Term "Biba Model" refers to this

Biba Implementation on LOCUS OS

- Goal: prevent untrusted software from altering data or other software
- 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
  - 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's Integrity Matrix Model

- First realistic commercial model
- Combines Bell-LaPadula, Biba models to obtain model conforming to requirements
  - Bell-LaPadula components
    - Security clearances: security level (audit, low) + category (devlp, prodcode, proddata,…)
  - Biba components
    - Integrity clearances: classification (system, operational, low) + category (devlp, prod)

Clark-Wilson Integrity Model

- 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* move system from one consistent state to another
- Issue: who examines, certifies transactions done correctly?

Entities

- CDIs: constrained data items
  - Data subject to integrity controls
- UDIs: unconstrained data items
  - Data not subject to integrity controls
- IVPs: integrity verification procedures
  - Procedures that test the CDIs conform to the integrity constraints
- TPs: transaction procedures
  - Procedures that take the system from one valid state to another

Certification Rules 1 and 2

CR1 When any IVP is run, it must ensure all CDIs are in a valid state

CR2 For some associated set of CDIs, a TP must transform those CDIs in a valid state into a (possibly different) valid state

- Defines relation *certified* that associates a set of CDIs with a particular TP
- Example: TP balance, CDIs accounts, in bank example

Enforcement Rules 1 and 2

ER1 The system must maintain the certified relations and must ensure that only TPs certified to run on a CDI manipulate that CDI.

ER2 The system must associate a user with each TP and set of CDIs. The TP may access those CDIs on behalf of the associated user. The TP cannot access that CDI on behalf of a user not associated with that TP and CDI.

- System must maintain, enforce certified relation
- System must also restrict access based on user ID (*allowed* relation)

Users and Rules

CR3 The allowed relations must meet the requirements imposed by the principle of separation of duty.

ER3 The system must authenticate each user attempting to execute a TP

- Type of authentication undefined, and depends on the instantiation
- Authentication *not* required before use of the system, but *is* required before manipulation of CDIs (requires using TPs)

Logging

CR4 All TPs must append enough information (to an append-only CDI) to reconstruct the operation.

- This CDI is the log
- Auditor needs to be able to determine what happened during reviews of transactions

Handling Untrusted Input

CR5 Any TP that takes as input a UDI may perform only valid transformations, or no transformations, for all possible values of the UDI. The transformation either rejects the UDI or transforms it into a CDI.

- In bank, numbers entered at keyboard are UDIs, so cannot be input to TPs. TPs must validate numbers (to make them a CDI) before using them; if validation fails, TP rejects UDI

Separation of Duty In Model

ER4 Only the certifier of a TP may change the list of entities associated with that TP. No certifier of a TP, or of an entity associated with that TP, may ever have execute permission with respect to that entity.

- Enforces separation of duty with respect to certified and allowed relations

Comparison: Requirement 1

"Users will not write their own programs, but will use existing production programs and databases."

- Users can't certify TPs: CR5 and ER4 enforce it

Comparison: Requirement 2

"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."

- 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

Comparison: Requirement 3

"A special process must be followed to install a program from the development system onto the production system."

- TP does the installation, trusted personnel do certification

Comparison: Requirement 4

"The special process in requirement 3 must be controlled and audited."

- CR4 provides logging; ER3 authenticates trusted personnel doing installation; CR5, ER4 control installation procedure
- New program UDI before certification, CDI (and TP) after

Comparison: Requirement 5

"The managers and auditors must have access to both the system state and the system logs that are generated."

- Log is CDI, so appropriate TP can provide managers, auditors access
- Access to state handled similarly

Comparison to Biba

- Biba
  - No notion of certification rules; trusted subjects ensure actions obey rules
  - Untrusted data examined before being made trusted
- Clark-Wilson
  - Explicit requirements that *actions* must meet
  - Trusted entity must certify *method* to upgrade untrusted data (and not certify the data itself)

Hybrid Policies

- Chinese Wall Model
  - Focuses on conflict of interest
- RBAC
  - Base controls on job function

Chinese Wall Model

Problem:

- Tony advises American Bank
- He is also asked to advise Toyland Bank

- Conflict of interest to accept, because his advice for either bank would affect his advice to the other bank

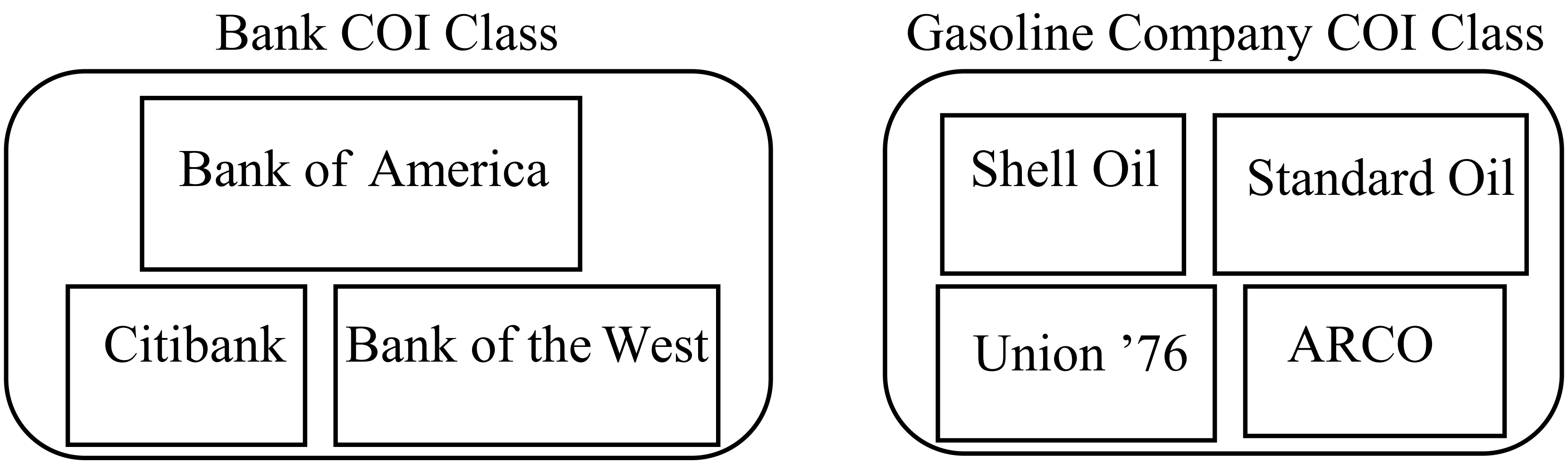
Organization

- Organize entities into "conflict of interest" classes
- Control subject accesses to each class
- Control writing to classes to ensure information flow is not violating rules
- Sanitized data can be viewed by everyone

Definitions

- *Objects*: items of information related to a company
- *Company dataset* (CD): contains objects related to a single company: $CD(O)$
- *Conflict of interest class* (COI): contains datasets of companies in competition: $COI(O)$
  - Assume: each object belongs to exactly one *COI* class

Example


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

Temporal Element

- If Anthony reads any CD in a COI, he can *never* read another CD in that COI
  - Possibility: information learned earlier may allow him to make decisions later !
  - $PR(S)$ = set of objects that S has already read

CW-Simple Security Condition

- s can read o iff either condition holds:
  1. There is an o' such that s has accessed o' and $CD(o') = CD(o)$

     (s has <u>already</u> read something in o's dataset)
  2. For all $o' \in PR(s) \Rightarrow COI(o') \neq COI(o)$

     (s has not read any objects in the conflict of interest class of o)
- Ignores sanitized data (see below)
- Initially, $PR(s) = \varnothing$, so initial read request granted

Sanitization

- 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*)
- Add option to CW-Simple Security Condition:
  3. *o* is a sanitized object

Writing

- Anthony, Susan work in same trading house
- Anthony: can read Bank1, Gas
- Susan: can read Bank2, Gas
- If Anthony could write to Gas then Susan could read it too !
  - Hence, indirectly, she can read information from Bank 1's CD, a clear conflict of interest

CW-*-Property

- *s* can write to *o* iff both of the following hold:
    1. The CW-simple security condition permits *s* to read *o*; and
    2. For all *unsanitized* objects *o*′, if *s* can read *o*′, then $CD(o') = CD(o)$
- Says that s can write to an object if all the (unsanitized) objects it can read are in the same dataset

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 emulate *current state* of CW only
- 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 also
- If "subjects" and "processes" are interchangeable, a single person could use multiple processes to violate CW-simple security condition
  - Would still comply with Clark-Wilson Model
- If "subject" is a specific person and includes all processes the subject executes, then consistent with Clark-Wilson Model

Role Based AC (RBAC)

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

Definitions

- Role r: collection of job functions
  - *trans*(r): set of authorized transactions for r
- Active role of subject s: role s is currently in
  - *actr*(s)
- Authorized roles of a subject s: set of roles s is authorized to assume
  - *authr*(s)
- *canexec*(s, t) iff subject s can execute transaction t at current time

Axioms

- Let S be the set of subjects and T the set of transactions.
- *Rule of role assignment*:
  $(\forall s \in S)(\forall t \in T)\ [canexec(s, t) \rightarrow actr(s) \neq \varnothing]$.
  - If s can execute a transaction, it has a role
  - This ties transactions to roles
- *Rule of role authorization*:
  $(\forall s \in S)\ [actr(s) \subseteq authr(s)]$.
  - Subject must be authorized to assume an active role (otherwise, any subject could assume any role)

Axiom

- *Rule of transaction authorization*:

  $$(\forall s \in S)(\forall t \in T)$$
  $$[canexec(s, t) \rightarrow t \in trans(actr(s))].$$

  – If a subject *s* can execute a transaction, then the transaction is an authorized one for the role *s* has assumed

Hierarchy of Roles

- Trainer (r) can do all transactions that trainee (r') can do (and then some). This means role r contains role r' ($r > r'$).
- Access to one role implies access to all roles containing it:

$$(\forall s \in S)[\ (r \in authr(s)) \wedge (r > r') \rightarrow r' \in authr(s)\ ]$$

Separation of Duty

- Let r be a role, and let s be a subject such that $r \in auth(s)$. Then the predicate $meauth(r)$ (for mutually exclusive authorizations) is the set of roles that s cannot assume because of the separation of duty requirement.
- Separation of duty: $(\forall r_1, r_2 \in R)$ $[\, r_2 \in meauth(r_1) \rightarrow [\, (\forall s \in S)\, [\, r_1 \in authr(s) \rightarrow r_2 \notin authr(s)\,]\,]\,]$

Key Points

- 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 based on multilevel integrity
  - Clark-Wilson focuses on separation of duty and transactions
- Hybrid policies deal with both confidentiality and integrity
  - Chinese Wall models conflicts of interest
  - RBAC model controls access based on functionality