Biba Integrity Model

Basis for all 3 models:

- 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
- $\underline{\mathrm{r}}$: $S \times O$ means $s \in S$ can read $o \in O$
- $\underline{\mathrm{w}}$, $\underline{\mathrm{x}}$ 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 $o_1, ..., o_{n+1}$ and a corresponding sequence of subjects $s_1, ..., s_n$ such that $s_i \underline{\text{r}}\, o_i$ and $s_i \underline{\text{w}}\, o_{i+1}$ 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 information transfer path from $o_1 \in O$ to $o_{n+1} \in O$, enforcement of low-water-mark policy requires $i(o_{n+1}) \leq i(o_1)$ for all $n > 1$.
  – Idea of proof: Assume information transfer path exists between o_1 and o_{n+1}. Assume that each read and write was performed in the order of the indices of the vertices. By induction, the integrity level for each subject is the minimum of the integrity levels for all objects preceding it in path, so $i(s_n) \leq i(o_1)$. As nth write succeeds, $i(o_{n+1}) \leq i(s_n)$. Hence $i(o_{n+1}) \leq i(o_1)$.

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 is 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

- Similar to Bell-LaPadula model
  1. $s \in S$ can read $o \in O$ iff $i(s) \leq i(o)$
  2. $s \in S$ can write to $o \in O$ iff $i(o) \leq i(s)$
  3. $s_1 \in S$ can execute $s_2 \in S$ iff $i(s_2) \leq i(s_1)$
- Add compartments and discretionary controls to get full dual of Bell-LaPadula model
- Information flow result holds
  - Different proof, though
- Term “Biba Model” refers to this

LOCUS and Biba

- 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

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 reconstruct the operation to an append-only CDI.

- 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 With Requirements

1. Users can't 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 controll installation procedure
   - New program UDI before certification, CDI (and TP) after
5. 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)

UNIX Implementation

- Considered “allowed” relation

  (*user*, *TP*, { *CDI set* })
- Each TP is owned by a different user
  - These “users” are actually locked accounts, so no real users can log into them; but this provides each TPO a unique UID for controlling access rights
  - TP is setuid to that user
- Each TP’s group contains set of users authorized to execute TP
- Each TP is executable by group, not by world

CDI Arrangement

- CDIs owned by *root* or some other unique user
  - Again, no logins to that user’s account allowed
- CDI’s group contains users of TPs allowed to manipulate CDI
- Now each TP can manipulate CDIs for single user

Examples

- Access to CDI constrained by user
  - In “allowed” triple, *TP* can be any TP
  - Put CDIs in a group containing all users authorized to modify CDI
- Access to CDI constrained by TP
  - In “allowed” triple, *user* can be any user
  - CDIs allow access to the owner, the user owning the TP
  - Make the TP world executable

Problems

- 2 different users cannot use same copy of TP to access 2 different CDIs
  - Need 2 separate copies of TP (one for each user and CDI set)
- TPs are setuid programs
  - As these change privileges, want to minimize their number
- *root* can assume identity of users owning TPs, and so cannot be separated from certifiers
  - No way to overcome this without changing nature of *root*

Chapter 7: Hybrid Policies

- Overview
- Chinese Wall Model
- Clinical Information Systems Security Policy
- ORCON
- RBAC

Overview

- Chinese Wall Model
  - Focuses on conflict of interest
- CISS Policy
  - Combines integrity and confidentiality
- ORCON
  - Combines mandatory, discretionary access controls
- RBAC
  - Base controls on job function

Chinese Wall Model

Problem:

- Tony advises American Bank about investments
- He is asked to advise Toyland Bank about investments

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

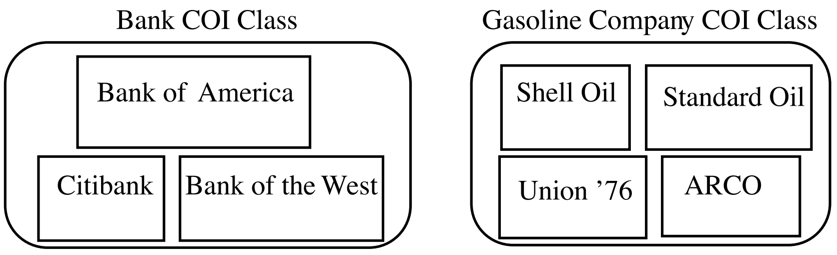
Organization

- Organize entities into “conflict of interest” classes
- Control subject accesses to each class
- Control writing to all classes to ensure information is not passed along in violation of rules
- Allow sanitized data to be viewed by everyone

Definitions

- *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



Temporal Element

- If Anthony reads any CD in a COI, he can *never* read another CD in that COI
  - Possible that information learned earlier may allow him to make decisions later
  - Let $PR(S)$ be 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 $CD(o') = CD(o)$
     - Meaning *s* has read something in *o*'s dataset
  2. For all $o' \in O$, $o' \in PR(s) \Rightarrow COI(o') \neq COI(o)$
     - Meaning *s* has not read any objects in *o*'s conflict of interest class
- Ignores sanitized data (see below)
- Initially, $PR(s) = \varnothing$, 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 third condition to CW-Simple Security Condition:
  3. *o* is a sanitized object

Writing

- Anthony, Susan work in same trading house
- Anthony can read Bank 1's CD, Gas' CD
- Susan can read Bank 2's CD, Gas' CD
- If Anthony could write to Gas' CD, Susan can read it
  - 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

Formalism

- Goal: figure out how information flows around system
- *S* set of subjects, *O* set of objects, $L = C \times D$ set of labels
- $l_1 : O \rightarrow C$ maps objects to their COI classes
- $l_2 : O \rightarrow D$ maps objects to their CDs
- *H*(*s*, *o*) true iff *s* has *or had* read access to *o*
- *R*(*s*, *o*): *s*'s request to read *o*

Axioms

- Axiom 7-1. For all $o, o' \in O$, if $l_2(o) = l_2(o')$, then $l_1(o) = l_1(o')$
  - CDs do not span COIs.
- Axiom 7-2. $s \in S$ can read $o \in O$ iff, for all $o' \in O$ such that $H(s, o')$, either $l_1(o') \neq l_1(o)$ or $l_2(o') = l_2(o)$
  - *s* can read *o* iff *o* is either in a different COI than every other *o´* that *s* has read, or in the same CD as *o*.

More Axioms

- Axiom 7-3. $\neg H(s, o)$ for all $s \in S$ and $o \in O$ is an initially secure state
  - Description of the initial state, assumed secure
- Axiom 7-4. If for some $s \in S$ and all $o \in O$, $\neg$H(s, o), then any request $R(s, o)$ is granted
  - If s has read no object, it can read any object

Which Objects Can Be Read?

- Suppose $s \in S$ has read $o \in O$. If s can read $o' \in O$, $o' \neq o$, then $l_1(o') \neq l_1(o)$ or $l_2(o') = l_2(o)$.
  - Says s can read only the objects in a single CD within any COI

Proof

Assume false. Then

$H(s, o) \wedge H(s, o') \wedge l_1(o') = l_1(o) \wedge l_2(o') \neq l_2(o)$

Assume s read o first. Then $H(s, o)$ when s read o, so by Axiom 7-2, either $l_1(o') \neq l_1(o)$ or $l_2(o') = l_2(o)$, so

$(l_1(\text{o}') \neq l_1(\text{o}) \vee l_2(o') = l_2(o)) \wedge (l_1(o') = l_1(o) \wedge l_2(o') \neq l_2(o))$

Rearranging terms,

$(l_1(o') \neq l_1(o) \wedge l_2(o') \neq l_2(o) \wedge l_1(o') = l_1(o)) \vee$

$(l_2(o') = l_2(o) \wedge l_2(o') \neq l_2(o) \wedge l_1(o') = l_1(o))$

which is obviously false, contradiction.

Lemma

- Suppose a subject $s \in S$ can read an object $o \in O$. Then s can read no o' for which $l_1(o') = l_1(o)$ and $l_2(o') \neq l_2(o)$.
  - So a subject can access at most one CD in each COI class
  - Proof sketch: Initial case follows from Axioms 7-3, 7-4. If $o' \neq o$, theorem immediately gives lemma.

COIs and Subjects

- Theorem: Let $c \in C$ and $d \in D$. Suppose there are n objects $o_i \in O$, $1 \leq i \leq n$, such that $l_1(o_i) = d$ for $1 \leq i \leq n$, and $l_2(o_i) \neq l_2(o_j)$, for $1 \leq i, j \leq n$, $i \neq j$. Then for all such o, there is an $s \in S$ that can read o iff $n \leq |S|$.
  - If a COI has n CDs, you need at least n subjects to access every object
  - Proof sketch: If s can read o, it cannot read any o' in another CD in that COI (Axiom 7-2). As there are n such CDs, there must be at least n subjects to meet the conditions of the theorem.

Sanitized Data

- $v(o)$: sanitized version of object o
  - For purposes of analysis, place them all in a special CD in a COI containing no other CDs
- Axiom 7-5. $l_1(o) = \mathrm{l}_1(v(o))$ iff $l_2(o) = l_2(v(o))$

Which Objects Can Be Written?

- Axiom 7-6. $s \in S$ can write to $o \in O$ iff the following hold simultaneously
  1. $H(s, o)$
  2. There is no $o' \in O$ with $H(s, o')$, $l_2(o) \neq l_2(o')$, $l_2(o) \neq l_2(v(o))$, $l_2(o') = l_2(v(o))$.
  - Allow writing iff information cannot leak from one subject to another through a mailbox
  - Note handling for sanitized objects

How Information Flows

- Definition: information may flow from *o* to *o´* if there is a subject such that $H(s, o)$ and $H(s, o')$.
  - Intuition: if *s* can read 2 objects, it can act on that knowledge; so information flows between the objects through the nexus of the subject
  - Write the above situation as (o, o')

Key Result

- Set of all information flows is
  $\{ (o,o') \mid o \in O \wedge o' \in O \wedge l_2(o) = l_2(o') \vee l_2(o) = l_2(v(o)) \}$
- Sketch of proof: Defn gives set of flows:
  $F = \{(o, o') \mid o \in O \wedge o' \in O \wedge \exists\, s \in S \text{ such that } H(s, o) \wedge H(s, o'))\}$

  Let F* be the transitive closure of this set. Axiom 7-6 excludes the following flows:
  $X = \{ (o, o') \mid o \in O \wedge o' \in O \wedge l_2(o) \neq l_2(o') \wedge l_2(o) \neq l_2(v(o)) \}$

  So
  $F^*{-}X = \{(o,o') \mid o \in O \wedge o' \in O \wedge \neg(l_2(o) \neq l_2(o') \wedge l_2(o) \neq l_2(v(o))) \}$

  which is equivalent to the claim.

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)
    - *S dom U*
  - Subjects assigned clearance for compartments without multiple categories corresponding to CDs in same COI class

May 4, 2004 ECS 235 Slide #47

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

May 4, 2004 ECS 235 Slide #48

Compare to Clark-Wilson

- Clark-Wilson Model covers integrity, so consider only access control aspects
- 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