Integrity Policies

- Overview
- Requirements
- Biba’s models
- Lipner’s model
- Clark-Wilson model

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 Policies

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.

Requirements Suggest Several Principles of Operation

Separation of Duty:

- It states that if two or more steps are required to perform a critical function, at least two different users should perform the steps.

Example of Critical Function: Moving a program from the development system to the production system.

Separation of Function:

- Developers do not develop new programs on production systems because of the potential threat to production data.
- Similarly, the developers do not process production data on the development systems.

Cont..

Auditing

- Commercial systems emphasize recovery and accountability.
- It is the process of analyzing systems to determine what actions took place and who performed them.
- Logging and auditing are especially important when programs move from the development system to the production system, since the integrity mechanisms typically do not constrain the certifier.

Information Aggregation

- By aggregating the innocuous information, one can often deduce much sensitive information.

Biba Integrity Model

Basis for all three models:

- A set of subjects S, a set of objects O, and a set of integrity levels I which are ordered.
- The relation $< \subseteq I \times I$ holds when second integrity level dominates first.
- The relation $\leq \subseteq I \times I$ holds when second integrity level dominates or is the same as the first.
- *The Function min*: $I \times I \rightarrow I$ returns the lesser of the two integrity levels with respect to $\leq$.
- *The Function i*: $S \cup O \rightarrow I$ returns the integrity level of an object or a subject.
- The relation r $\subseteq S \times O$ defines the ability of a subject $s \in S$ to read an object $o \in O$.
- The relation w $\subseteq S \times O$ defines the ability of a subject $s \in S$ to write an object $o \in O$.
- The relation ex $\subseteq S \times \mathrm{S}$ defines the ability of a subject $s_1 \in S$ to invoke another subject $s_2 \in S$.

Intuition for Integrity Levels

- The higher the level, the more confidence one has
  - that a program will execute correctly.
  - that data is accurate and/or reliable than data at lower level.
- Trustworthiness is used as a measure of integrity level.
- Important point: integrity levels (prevent modification of information) are **not** security levels (prevent flow of information).

Biba Tests Policies Against Notion of an 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 r o_i and s_i w o_{i+1} for all i, $1 \le i \le n$.
- Intuitively, information can flow from o_1 to o_{n+1} along an information flow path by a successive reads and writes.

Low-Water-Mark Policy

- Idea: When s accesses o, the policy changes the integrity level of the subject to the lower of *the subject and the object*.
- 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 $o_{n+1} \in O$, then enforcement of the low-water-mark policy requires that $i(o_{n+1}) \leq i(o_1)$ for all $n > 1$.
  - Idea of proof: Assume information transfer path exists between o_1 and o_{n+1}.
  - Also assume that each read and write was performed in the order of the indices of the vertices.
  - By induction, for any $1 \leq k \leq n$,

$$i(s_k) = \min \{i(o_j) \mid 1 \leq j \leq k\} \text{ after } k^{th} \text{ read}$$

Cont..

- As nth write succeeds, $i(o_{n+1}) \le i(s_n)$.
- The integrity level for each subject is the minimum of the integrity levels for all objects preceding it in path, so $i(s_n) \le i(o_1)$.
- Thus by transitivity, $i(o_{n+1}) \le i(o_1)$.

Issue

- 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.
- The crux of issue is model prevents indirect modification.
  - Because subject levels lowered when subject reads from low-integrity object.

Ring Policy

- Idea: Ignore the issue of indirect modification and focuses on direct modification. Subject integrity levels static.
- Rules
  1. Any subject can read any object.
  2. $s \in S$ can write to $o \in O$ if and only if $i(o) \leq i(s)$.
  3. $s_1 \in S$ can execute $s_2 \in S$ if and only if $i(s_2) \leq i(s_1)$.
- 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 integrity compartments and discretionary controls to get full dual of Bell-LaPadula model.
- Information flow result holds.
- Prevents indirect as well as direct modifications.

Integrity Matrix Model

- Lipner proposed this as first realistic commercial model.
- Combines Bell-LaPadula and Biba models to obtain model conforming to requirements.
- Do it in two steps.
  - Bell-LaPadula components first.
  - Then add Biba's components.

Bell-LaPadula Clearances

- Lipner provides two security levels, in the following orders (higher to lower):
  - AM (Audit Manager): System audit and management functions at this level.
  - SL (System Low): Any process can read information at this level.

Bell-LaPadula Categories

- Five categories:
  - D (Development): Production programs under development and testing, but not yet in production use.
  - PC (Production Code): Production processes and programs.
  - PD (Production Data): Data covered by integrity policy.
  - SD (System Development): System programs under development, but not yet in production use.
  - T (Software Tools): Programs provided on the production system not related to the sensitive or protected data.

Users and Security Levels

Subjects	Security Level
Ordinary users	(SL, { PC, PD })
Application developers	(SL, { D, T })
System programmers	(SL, { SD, T })
System managers and auditors	(AM, { D, PC, PD, SD, T })
System controllers	(SL, {D, PC, PD, SD, T}) and downgrade privilege

Objects and Classifications

Objects	**Security Level**
Development code/test data	(SL, { D, T })
Production code	(SL, { PC })
Production data	(SL, { PC, PD })
Software tools	(SL, { T })
System programs	(SL, $\varnothing$)
System programs in modification	(SL, { SD, T })
System and application logs	(AM, { *appropriate* })

Ideas

- Ordinary users can execute (read) production code but cannot alter it.
- Ordinary users can alter and read production data.
- System managers need access to all logs but cannot change levels of objects.
- System controllers need to install code (hence downgrade capability).
- Logs are append only, so must dominate subjects writing them.

Check Requirements

1. Users have no access to T, so cannot write their own programs.
2. Applications programmers have no access to PD, so cannot access production data; if needed, it must be put into D, requiring the system controller to intervene.
3. Installing a program requires downgrade procedure (from D to PC), so only system controllers can do it.

More Requirements

4. Control: only system controllers can downgrade; audit: any such downgrading must be altered.
5. 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
    - System managers at AM, production data at SL
- So add more …

Adding Biba

- Three integrity classifications (highest to lowest):
  - ISP(System Program): For system programs
  - IO (Operational): For production programs and development software.
  - ISL (System Low): Users get this on log in.
- Two integrity categories:
  - ID (Development): Development entities.
  - IP (Production): Production entities.

Simplify Bell-LaPadula

- Reduce security categories to three:
  - SP (Production): Production code and data.
  - SD (Development): Same as security category D.
  - SSD (System Development): Same as security category SD.

Users and Levels

Subjects	Security Level	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 })

Objects and Classifications

Objects	Security Level	Integrity Level
Development code/test data	(SL, { SD })	(ISL, { IP})
Production code	(SL, { SP })	(IO, { IP })
Production data	(SL, { SP })	(ISL, { IP })
Software tools	(SL, $\varnothing$)	(IO, { ID })
System programs	(SL, $\varnothing$)	(ISP, { IP, ID })
System programs in modification	(SL, { SSD })	(ISL, { ID })
System and application logs	(AM, { *appropriate* })	(ISL, $\varnothing$)
Repair	(SL, {SP})	(ISL, { IP })

Ideas

- 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.
  - Note writing constraints removed from security classes.

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

Entities

- CDIs: Constrained Data Items
  - Data subject to integrity controls (Ex.- Account balance)
- UDIs: Unconstrained Data Items
  - Data not subject to integrity controls (Ex.- Gift at the time of account opening)
- IVPs: Integrity Verification Procedures
  - Procedures that test the CDIs conform to the integrity constraints. (Ex.- Verifying account balance)
- TPs: Transaction Procedures
  - Procedures that take the system from one valid state to another. (Ex.- Deposit Money, Withdraw Money and Transfer Money.)

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 a 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).
- This defines a set of triples (*user, TP,* { *CDI set* }) to capture the association of users, TPs, and CDIs.

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 is *not* required when a user logs into 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.
- An 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.

Contribution

The model contributes two new ideas to integrity models.

1. Firms don't classify data using a multilevel scheme and they ensure separation of duty.
2. Notion of certification is distinct from the notion of enforcement, and each has its own set of rules.

Comparison With Requirements

1. Ordinary users can't write program to access production databases. They must use existing TPs and CDIs (that is production program and production databases), so CR5 and ER4 enforce this
2. Procedural, 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 and ER4 control installation procedure.
   - New program UDI before certification, CDI (and TP) after.
5. Log is CDI, so appropriate TP can provide managers and 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 TP 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

- Two different users cannot use same copy of TP to access two different CDIs
  - Need two 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*\

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, Lipner based on multilevel integrity.
- Clark-Wilson focuses on separation of duty and transactions.