Integrity Policies

Murat Kantarcioglu

Requirements of Policies for Commercial Applications [Lipner 1982]

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.

The emphasis of these requirements is on integrity.

Requirements and Principles of Operation

- The requirements suggest several principles
  - *Separation of duty.* If two or more steps are required to perform a critical function, at least two different subjects should perform them. Moving an application from the development system to the production system is an example of critical function.
  - *Separation of function.* Different functions are executed on different sets of data. For example, developers do not develop new programs in the production environment. Also they do not process production data in the development environment. If they need data, depending on the sensitivity of data, sanitized versions of these data may be given to them.
  - *Auditing*. It is the process of analyzing systems to determine what actions took place and who performed them

Biba Integrity Model

- The Biba model associates an integrity level with both objects and subjects
- These levels form the basis for expressing integrity policies that refer to the corruption of ‘clean’ high level entities by ‘dirty’ low level entities
- In the integrity lattice, information may only flow downwards

Biba Integrity Model

- Set of subjects S, objects O, integrity levels I, relation $\leq \subseteq I \times I$ holding when the second dominates the first
- i: $S \cup O \rightarrow I$ gives integrity level of an object or of a subject
- $\underline{r} \subseteq S \times O$ means $s \in S$ can read $o \in O$
- $\underline{w} \subseteq S \times O$ means $s \in S$ can write $o \in O$
- $\underline{x} \subseteq S \times S$ defines the ability of a subject to invoke (execute) another subject.

Intuition for Integrity Levels

- The higher the level, the more confidence
  - That a program will execute correctly
  - That data is accurate and/or reliable
- Important point: *integrity levels are **not** security levels*
- Integrity labels are assigned and maintained separately, because the reasons behind the labels are different

Biba's Model

- Access Control Rules
  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 $i(s_2) \leq i(s_1)$
- No actual implementations in real products have been reported for the Biba model
- The problem of integrity of information requires articulated solutions

Clark-Wilson Integrity Model

- This model is based on two important principles:
  - Separation of duties
  - Well-formed transactions – these transactions constrain the ways in which users can modify the data. The main idea is that a data item can be modified only by a given set of transactions that are certified to work with that data item
- Unlike the Bell-LaPadula security model, which relies on access mediation in the operating system kernel (or DBMS), Clark and Wilson's approach relies on application-level controls

Clark-Wilson Integrity Model

- Main points of the Clark-Wilson model
  1. Subjects have to be identified and authenticated
  2. Objects can be manipulated only by a restricted set of programs
  3. Subjects can execute only a restricted set of programs
  4. A proper audit log has to be maintained
  5. The system has to be certified to work properly

Clark-Wilson Integrity Model

- Integrity defined by a set of constraints
  - Data in a *consistent* or valid state when it satisfies these constraints
- Example: Bank
  - *D* today's deposits, *W* withdrawals, *YB* yesterday's balance, *TB* today's balance
  - Integrity constraint: $D + YB - W$
- *Well-formed transactions* 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 that 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

- CR2 defines as *certified* a relation that associates a set of CDIs with a particular TP
- *CR2 implies that a TP may corrupt a CDI if it is not certified to work with that CDI*
- Bank example: TP balance, CDIs accounts. Let C be a certified relation, then

  (balance, $account_1$), (balance, $account_2$),….,

  (balance, $account_n$) $\in$ C

Enforcement Rules 1 and 2

- *CR2 implies that a TP may corrupt a CDI if it is not certified to work with that CDI*
- Example: the TP that invests money in the bank's stock portfolio would corrupt account balances even if the TP were certified to work on the portfolio, because the actions of the TP make no sense on the bank account
- This leads to the first enforcement rule
- The second enforcement rule is motivated by the fact that not all users are allowed to use all the TPs; the model must thus also account for the person performing the TP

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)

Enforcement Rule 2 – allowed relation

- The allowed relation *A* specifies which user execute which TP on which CDI
- Let *U* be the set of users in the system

  Let *T* be the set of TP's in the system

  Let *C* be the set of CDI's in the system

  A is defined as:

  $\{<u, tp, cdi\_s> \mid u \in U, tp \in T, cdi\_s \in 2^C\}$

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. TPs must validate numbers (to make them a CDI) before using them; if validation fails, TP rejects UDI
- Therefore CR5 says that any TP that takes a UDI as input must either convert the UDI into a CDI or reject the UDI and perform no transformation at all

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 cannot 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 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)

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