Integrity Policies

- Overview
- Biba's models
  - Strict Integrity policy
  - Ring policy
  - Low-Water-Mark policy
- Lipner's requirements
- Clark-Wilson model

1

Background and Overview

- Commercial world different from military world
  - The focus is on integrity (can IT be trusted) rather then confidentiality (can IT be divulged)
- Integrity levels used to label subjects and objects
  - The higher the level, the more trustworthy the subject (behavior) or the object (contents)
- Integrity level ≠ confidentiality levels
  - Subjects with Top Secret clearance are also trusted
  - A system binary is trusted but not secret (any user can read)
  - An untrusted Java applet is secret (only admin can read=execute)
- Information lows from trusted to untrusted
  - Don't want (trusted) IE to open (untrusted) JPG file with virus
  - Don't want (untrusted) downloaded Java applet to write into (trusted) Windows registry

2

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 or the same (linear or p.o.)
- *min*: $I \times I \rightarrow I$ returns lesser of integrity levels
- $i : S \cup O \rightarrow I$ gives integrity level of entity
- $\underline{r} \subseteq S \times O$ defines which $s \in S$ can read $o \in O$
- Similarly, $s \underline{w} o$, $s \underline{x} o$ indicate that s can write o and s can execute o

3

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*

4

Strict Integrity Policy

Similar to Bell-LaPadula model

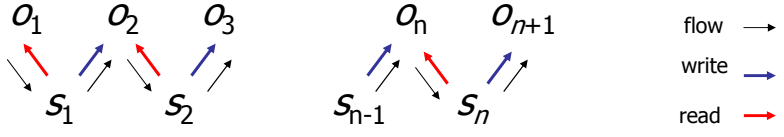
- For all $s \in S$ and $o \in O$
  1. s r o iff $i(s) \leq i(o)$ (read up)
  2. s w o iff $i(o) \leq i(s)$ (write down)
  3. s_1 x s_2 iff $i(s_2) \leq i(s_1)$ (execute down)

- Term "Biba Model" refers to this

5

Information Transfer Path

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



6

Information Flow and Model

- If there is an information transfer path from $o_1 \in O$ to $o_{n+1} \in O$, enforcement of the strict 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 $i(o_{n+1}) \leq i(o_1)$.

7

LOCUS and (strict) Biba

- Goal: prevent untrusted software from altering data or other software in distributed Op.Sys.
- Approach: make levels of trust explicit
  - *credibility rating* (Biba's levels) based on estimate of software's trustworthiness (0 untrusted, n highly trusted) based on source of software
  - *trusted file systems* contain software with a single credibility level
  - Process has associated *risk level* starting at highest credibility level; process can execute programs with no lower credibility
  - Must use *run-untrusted* command to run software at lower credibility level (acknowledging risk)

8

Ring Policy

Only concerned about which subjects can directly modify objects

- 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 $I(s_2) \leq I(s_1)$.
- Ignores indirect modification problem
- Information flow result does NOT hold (!)

9

Low-Water-Mark Policy

Idea: relax strict and ring integrity constraints, while maintaining Information Flow result

- Two versions
  1. Subject Low-Water-Mark policy relaxes the read constraints allowing subjects to read down.
  2. Object Low-Water-Mark policy relaxes the write constraints allowing subjects to write up.

10

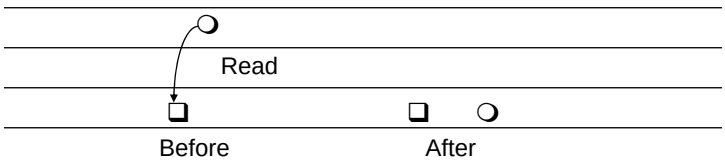
Subject Low-Water-Mark Policy

- Idea: *s* can read down, but its integrity level is 'tainted' by *o*'s integrity level
  - A 'trusted' program using untrusted input produces untrusted output
  - After reading a virus infected email message, the system can no longer be trusted and should be isolated
- Rules
  1. $s \underline{w} o$ if and only if $i(o) \leq i(s)$.
  2. If $s \in S$ reads $o \in O$, then the subject's new integrity level is $i'(s) = min(i(s), i(o))$.
  3. $s_1 \underline{x} s_2$ if and only if $i(s_2) \leq i(s_1)$.

11

Subject Low-Water Mark

- Relax NRD: Read down is allowed, but:
  - It lowers the reading subject to the low object's level
  - The subject is "tainted" with the object's integrity level
- No restriction on what a subject can read
  - If the subject can live with the consequences
- Integrity level of subjects is *monotonic*
  - Either stays the same, or drops



12

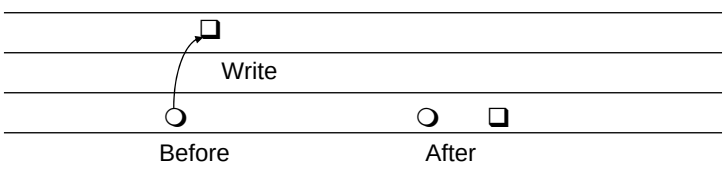
Object Low-Water-Mark Policy

- Idea: *s* can write up, but the integrity level of the written object is 'tainted' by *s*'s level
  - An 'untrusted' program produces untrusted output
  - Any file modified by a virus-infected program can no longer be trusted and should be deleted
- Rules
  1. $s \underline{r}\ o$ if and only if $i(s) \leq i(o)$.
  2. If $s \in S$ writes $o \in O$, then the object's new integrity level is $i'(o) = min(i(s), i(o))$.
  3. $s_1 \underline{x}\ s_2$ if and only if $i(s_2) \leq i(s_1)$.

13

Object Low-Water Mark

- Relax NWU: Write up is allowed, but:
  - It lowers the written object to the low subject's level
  - The object is "tainted" with the subject's integrity level
- so "low" can corrupt anything, that just gets recorded as corrupted (!?!)



14

Problems

- With Subject-LWM policy, the subjects' integrity levels never increase as system runs
  - Soon no subject will be able to access objects at high integrity levels
- With Object-LWM policy, objects can be accessed and easily corrupted
  - Soon all objects will be at the lowest integrity level
- A mechanism is needed in an implementation to warn about corruption of subjects and objects

15

Biba Summary

- Biba is actually 3 models: S/O-LWM, ring, strict
- Strict Biba has some practical limitations
  - Implemented on distributed operating system LOCUS
- No mechanism or procedure to verify trusted subject's actions (passing input from uncontrolled sources to higher level)
- Low water mark policy implemented in LOMAC (http://opensource.nailabs.com/lomac/), a security module for Free UNIX kernels

16

Requirements of Policies [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.

17

Clark-Wilson Integrity Model

- Radically different integrity model [1987]
- Motivated by 'paper' integrity practices, i.e. accounting rules (in place since 1960's)
- Integrity requirements divided into
  - Internal consistency: referring to properties of the internal state, enforced by the system
  - External consistency: referring to the relation of internal state with real world, enforced outside the system (auditing)
- Basic idea:
  - High integrity objects may only be manipulated using ***well-formed transactions*** (no direct access)
- Issue:
  - Who examines and certifies transactions done correctly? Who certifies certifiers and monitors auditors?

18

Clark-Wilson Integrity Model

- Integrity defined by a set of constraints
  - Data is in a *consistent* or *valid* state when it satisfies these constraints
- Example: Bank Account
  - today's deposits *D* - today's withdrawals *W* + yesterday's balance *YB* = today's balance *TB*
- *Well-formed transaction* move system from one consistent state to another
- Hierarchical structure
  - Level 1: transactions must satisfy this constraint
  - Level 2: users can only use these transactions
  - Level 3: certifiers ensure requirements at level 1 and 2
  - Level 4: logs monitor that certifiers do so

19

Entities

- Users: active agents
- CDIs: constrained data items
  - Data subject to integrity controls (balance in account)
  - Manipulated only by TPs
- UDIs: unconstrained data items
  - Data not subject to integrity controls
  - May be manipulated by users via primitive write operations
- IVPs: integrity verification procedures
  - Procedures that periodically test that the CDIs conform to the integrity constraints (consistency condition)
- TPs: transaction procedures
  - Procedures that take the system from one valid state to another (deposit/withdraw/transfer money)

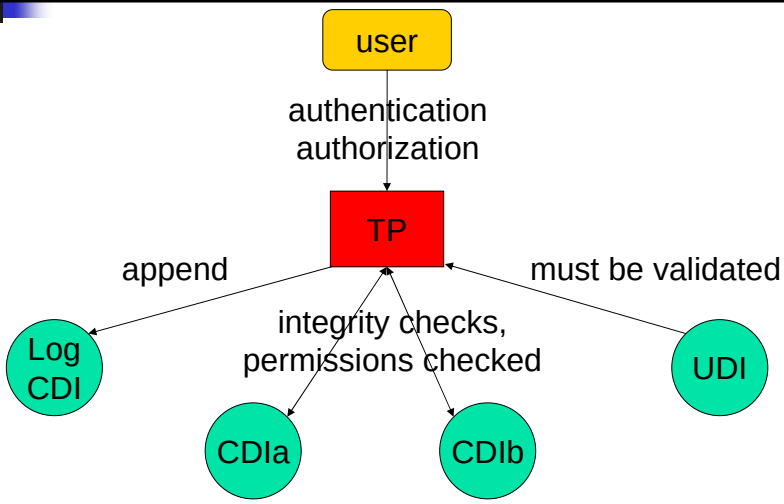
20

Clark-Wilson Rules

CR1	IVPs verify CDI integrity in a state
CR2	TPs preserve CDI integrity (valid state)
CR3	Suitable (static) separation of duties (for ER2)
CR4	TPs write to log
CR5	TPs upgrade UDIs to CDIs
ER1	CDIs changed only by authorized TP
ER2	Users only use authorized TP and CDI
ER3	Users are authenticated
ER4	Authorizations changed only by certifiers

21

Access Control in Clark-Wilson



22

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 withdraw maintains the balance as certified for CDIs accounts, in bank example
- No guarantees on other CDIs

23

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)

24

Users and Rules

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

(and this needs:)

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 (to manipulate UDIs), but *is* required before manipulation of CDIs (requires using TPs)

25

Logging

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

- This CDI is the log
- No TP can overwrite the log
- Auditor needs to be able to determine what happened during reviews of transactions

26

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 (deposit?) 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 (money in envelope does not match number), TP rejects UDI

27

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 on that TP.

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

28

Comparison With Requirements

1. Users will not write their own programs, but will use existing production programs and databases.
   - Users cannot certify TPs, so CR5 and ER4 enforce this
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 does not directly cover it; special process corresponds to using TP (e.g., to sanitize production data)
     - No technical controls can prevent programmer from developing program on production system; usual control is to delete software tools

29

Comparison With Requirements

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
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 the certification

30

Comparison With Requirements

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

SO Lipner's requirements met by Clark-Wilson

31

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
- Note the difference between a general purpose operating system (BLP/Biba) and an application oriented IT system (Clark-Wilson).

32