Integrity Policies
CS3SR3/SE3RA3

Ryszard Janicki

Acknowledgments: Material based on Computer Security: Art and Science by Matt Bishop (Chapter 6)
Outline

1. Integrity Policies

2. The Biba Integrity Model
Definition (Integrity)

Trustworthiness of data or resources. Integrity policies focus on integrity rather than confidentiality, as most commercial and industrial firms are more concerned with accuracy than disclosure.

- Commercial requirements differ from military requirements in their emphasizes on preserving data integrity.
Goals & 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.
Principles of Operation

- **SEPARATION OF DUTY**: If two or more steps are required to perform a *critical function*, at least two different people should perform the steps.

  **Example**
  Critical function = moving a program from the development system to the production system.

- **SEPARATION OF FUNCTION**: Do not develop new programs on existing production systems. Do not process real production data on development system. Development environment and actual production environment should be as similar as possible.

- **AUDITING**: The process of analyzing systems to determine what actions took place and who performed them. Commercial systems emphasize recovery and accountability.
Definition (Biba Model)

A system consists of a set $S$ of subjects, a set $O$ of objects, and a set $I$ of integrity levels. For each $i_1, i_2 \in I$, we say $i_1 < i_2$ if $i_2$ dominates $i_1$. We assume $<$ is a total order (for each $i_1, i_2$: $i_1 < i_2 \land i_2 < i_1 \land i_1 = i_2$).

- The function $\min(i_1, i_2)$ gives the smaller (w.r.t. $<$) element.
- 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 to read an object.
- The relation $w \subseteq S \times O$ defines the ability of a subject to write to an object.
- The relation $x \subseteq S \times O$ defines the ability of a subject to invoke (execute) another object.
Intuition of Biba Integrity Model

- The higher the level, the more confidence one has that a program will execute correctly, or detect problems with its inputs and stop executing.
- Data at a higher level is more accurate and/or reliable (with respect to some metric) than data on lower level.
- **TRUST**: a process at a higher level than that of an object is considered more *trustworthy* than that object.

**Integrity labels, in general, are not also security labels.**
- **SECURITY LABELS**: limit the flow of information,
- **INTEGRITY LABELS**: inhibit the modification of information.

There are three versions of Biba’s models: *Low-Water-Mark*, *Ring* and *Strict Integrity*. 

An Information Transfer Path is a sequence of objects $o_1, \ldots, o_{n+1}$ and a corresponding sequence of subjects $s_1, \ldots, s_n$ such that $s_i \text{ r } o_i$ and $s_i \text{ w } o_{i+1}$, for all $i = 1, \ldots, n$.

Intuitively, data in the object $o_1$ can be transferred into the object $o_{n+1}$ along an information flow pathy by a succession of reads and writes.
Low Water Mark Policy

Definition

Whenever a subject accesses an object, the policy changes the integrity level of the subject to the lower of the subject and the object. Specifically:

1. $s \in S$ can write $o \in O \iff i(o) \leq i(s)$,
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 $s_2 \in S \iff i(s_2) \leq i(s_1)$.

- Rule (1) prevents writing to a higher level. This prevents a subject from writing to a more highly trusted object.
- Rule (2) causes a subject’s integrity level to drop whenever it reads an object at a lower integrity level.
- Rule (3) allows a subject to execute another subject provided the second is not at a higher integrity level.
Low Water Mark Policy: Motivations

1. Number one prevents a subject from writing to a more highly trusted object. Intuitively, if a subject were to alter a more trusted object, it could implant incorrect or false data (because the subject is less trusted than the object). In some sense, the trustworthiness of the object would drop to that of the subject. Hence such writing is disallowed.

2. The idea is that the subject is relying on data less trustworthy than itself. Hence, its trustworthiness drops to the lesser trustworthy level. This prevents data from “contaminating” the subject or its action.

3. Otherwise, the less trusted invoker could control the execution of the invoked object, corrupting it even though it is more trustworthy.
Low Water Mark Policy

Theorem (Policy Constraints)

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

- This policy prevents direct modifications that lower integrity labels.
- It also prevents indirect modification by lowering the integrity label of a subject that reads from an object with a lower integrity level.
- The problem with this policy is that, in practice, the subjects change integrity levels.
- In particular, the level of a subject is non-increasing, which means that it will soon be unable to access objects at a high integrity level.
- An alternative policy is to decrease object integrity levels rather than subject integrity levels, but this policy has the property of downgrading object integrity levels to the lowest level.
The Ring Policy ignores the issue of indirect modification and focuses on direct modification only.

It solves the problems from the previous page!

Definition (Ring Policy)

1. Any subject may read any object, regardless of integrity level.
2. $s \in S$ can write $o \in O \iff i(o) \leq i(s)$.
3. $s_1 \in S$ can execute $s_2 \in S \iff i(s_2) \leq i(s_1)$.

The difference between this policy and the low water mark policy is that any subject can read any object.
Strict Integrity Policy

This model is dual to the Bell-LaPadula Model.

**Definition (Strict Integrity Policy)**

1. $s \in S$ can read $o \in O \iff i(s) \leq i(o)$.
2. $s \in S$ can write $o \in O \iff i(o) \leq i(s)$.
3. $s_1 \in S$ can execute $s_2 \in S \iff i(s_2) \leq i(s_1)$.

- The Policy Constraint Theorem still holds.
- Rules (1) and (2) imply that if both *read* and *write* are allowed: $i(o) = i(s)$.
- Like the low water mark policy, this policy prevents indirect as well as direct modification of entities without authorization.
- By replacing the notion of *integrity level* with *integrity compartments*, and adding the notion of discretionary control, one obtains the full dual of Bell-LaPadula.