6.2. Biba Integrity Model

In 1977, Biba [94] studied the nature of the integrity of systems. He proposed three policies, one of which was the mathematical dual of the Bell-LaPadula Model.

A system consists of a set $\mathbf{S}$ of subjects, a set $\mathbf{O}$ of objects, and a set $\mathbf{I}$ of integrity levels.[1] The levels are ordered. The relation $< \subseteq \mathbf{I} \times \mathbf{I}$ holds when the second integrity level dominates the first. The relation $\leq \subseteq \mathbf{I} \times \mathbf{I}$ holds when the second integrity level either dominates or is the same as the first. The function $\mathbf{min}: \mathbf{I} \times \mathbf{I} \rightarrow \mathbf{I}$ gives the lesser of the two integrity levels (with respect to $\leq$). The function $\mathbf{i}: \mathbf{S} \cup \mathbf{O} \rightarrow \mathrm{I}$ returns the integrity level of an object or a subject. The relation $\underline{r} \subseteq \mathbf{S} \times \mathbf{O}$ defines the ability of a subject to read an object; the relation $\underline{w} \subseteq \mathbf{S} \times \mathbf{O}$ defines the ability of a subject to write to an object; and the relation $\underline{x} \subseteq \mathbf{S} \times \mathbf{S}$ defines the ability of a subject to invoke (execute) another subject.

[1] The original model did not include categories and compartments. The changes required to add them are straightforward.

Some comments on the meaning of "integrity level" will provide intuition behind the constructions to follow. 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 at a lower level. Again, this model implicitly incorporates the notion of "trust"; in fact, the term "trustworthiness" is used as a measure of integrity level. For example, a process at a level higher than that of an object is considered more "trustworthy" than that object.

Integrity labels, in general, are not also security labels. They are assigned and maintained separately, because the reasons behind the labels are different. Security labels primarily limit the flow of information; integrity labels primarily inhibit the modification of information. They may overlap, however, with surprising results (see Exercise 3).

Biba tests his policies against the notion of an information transfer path:

> **Definition 6–1.** An **information transfer path** is a sequence of objects $\mathbf{o}_1, \ldots, \mathbf{o}_{n+1}$ and a corresponding sequence of subjects $\mathbf{s}_1, \ldots, \mathbf{s}_n$ such that $\mathbf{s}_i \,\underline{r}\, \mathbf{o}_i$ and $\mathbf{s}_i \,\underline{w}\, \mathbf{o}_{i+1}$ for all $\mathbf{i}$, $1 \leq \mathbf{i} \leq \mathbf{n}$.

Intuitively, data in the object $\mathbf{o}_1$ can be transferred into the object $\mathbf{o}_{n+1}$ along an information flow path by a succession of reads and writes.

6.2.1. Low-Water-Mark Policy

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. $\mathbf{s} \in \mathbf{S}$ can write to $\mathbf{o} \in \mathbf{O}$ if and only if $\mathbf{i}(\mathbf{o}) \leq \mathbf{i}(\mathbf{s})$.
2. If $\mathbf{s} \in \mathbf{S}$ reads $\mathbf{o} \in \mathbf{O}$, then $\mathbf{i}'(\mathbf{s}) = \mathbf{min}(\mathbf{i}(\mathbf{s}), \mathbf{i}(\mathbf{o}))$, where $\mathbf{i}'(\mathbf{s})$ is the subject's integrity level after the read.

3. $\mathbf{s}_1 \in \mathbf{S}$ can execute $\mathbf{s}_2 \in \mathbf{S}$ if and only if $\mathbf{i}(\mathbf{s}_2) \leq \mathbf{i}(\mathbf{s}_1)$.

The first rule prevents writing from one level to a higher level. This 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.

The second rule causes a subject's integrity level to drop whenever it reads an object at a lower integrity level. 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 the data from "contaminating" the subject or its actions.

The third rule allows a subject to execute another subject provided the second is not at a higher integrity level. Otherwise, the less trusted invoker could control the execution of the invoked subject, corrupting it even though it is more trustworthy.

This policy constrains any information transfer path:

Theorem 6–1. If there is an information transfer path from object $\mathbf{o}_1 \in \mathbf{O}$ to object $\mathbf{o}_{n+1} \in \mathbf{O}$, then enforcement of the low-water-mark policy requires that $\mathbf{i}(\mathbf{o}_{n+1}) \leq \mathbf{i}(\mathbf{o}_1)$ for all $\mathbf{n} > 1$.

Proof If an information transfer path exists between $\mathbf{o}_1$ and $\mathbf{o}_{n+1}$, then Definition 6–1 gives a sequence of subjects and objects identifying the entities on the path. Without loss of generality, assume that each read and write was performed in the order of the indices of the vertices. By induction, for any $1 \leq \mathbf{k} \leq \mathbf{n}$, $\mathbf{i}(\mathbf{s}_k) = \mathbf{min}\{\, \mathbf{i}(\mathbf{o}_j) \mid 1 \leq \mathbf{j} \leq \mathbf{k} \,\}$ after $\mathbf{k}$ reads. As the $\mathbf{n}$th write succeeds, by rule 1, $\mathbf{i}(\mathbf{o}_{n+1}) \leq \mathbf{i}(\mathbf{s}_n)$. Thus, by transitivity, $\mathbf{i}(\mathbf{o}_{n+1}) \leq \mathbf{i}(\mathbf{o}_1)$.

This policy prevents direct modifications that would 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 nonincreasing, 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.

6.2.2. Ring Policy

The ring policy ignores the issue of indirect modification and focuses on direct modification only. This solves the problems described above. The rules are as follows.

1. Any subject may read any object, regardless of integrity levels.
2. $\mathbf{s} \in \mathbf{S}$ can write to $\mathbf{o} \in \mathbf{O}$ if and only if $\mathbf{i}(\mathbf{o}) \leq \mathbf{i}(\mathbf{s})$.
3. $\mathbf{s}_1 \in \mathbf{S}$ can execute $\mathbf{s}_2 \in \mathbf{S}$ if and only if $\mathbf{i}(\mathbf{s}_2) \leq \mathbf{i}(\mathbf{s}_1)$.

The difference between this policy and the low-water-mark policy is simply that any subject can read any object. Hence, Theorem 6–1 holds for this model, too.

6.2.3. Biba's Model (Strict Integrity Policy)

This model is the dual of the Bell-LaPadula Model, and is most commonly called “Biba's model.” Its rules are as follows.

1. $\mathbf{s} \in \mathbf{S}$ can read $\mathbf{o} \in \mathbf{O}$ if and only if $\mathbf{i(s)} \leq \mathbf{i(o)}$.
2. $\mathbf{s} \in \mathbf{S}$ can write to $\mathbf{o} \in \mathbf{O}$ if and only if $\mathbf{i(o)} \leq \mathbf{i(s)}$.
3. $\mathbf{s}_1 \in \mathbf{S}$ can execute $\mathbf{s}_2 \in \mathbf{S}$ if and only if $\mathbf{i}(\mathbf{s}_2) \leq \mathbf{i}(\mathbf{s}_1)$.

Given these rules, Theorem 6–1 still holds, but its proof changes (see Exercise 1). Note that rules 1 and 2 imply that if both read and write are allowed, $\mathbf{i(s)} = \mathbf{i(o)}$.

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 controls, one obtains the full dual of Bell-LaPadula.

EXAMPLE: Pozzo and Gray [817, 818] implemented Biba's strict integrity model on the distributed operating system LOCUS [811]. Their goal was to limit execution domains for each program to prevent untrusted software from altering data or other software. Their approach was to make the level of trust in software and data explicit. They have different classes of executable programs. Their **credibility ratings** (Biba's integrity levels) assign a measure of trustworthiness on a scale from 0 (untrusted) to **n** (highly trusted), depending on the source of the software. Trusted file systems contain only executable files with the same credibility level. Associated with each user (process) is a **risk level** that starts out set to the highest credibility level at which that user can execute. Users may execute programs with credibility levels at least as great as the user's risk level. To execute programs at a lower credibility level, a user must use the **run-untrusted** command. This acknowledges the risk that the user is taking.
