6.3. Lipner's Integrity Matrix Model

Lipner returned to the Bell-LaPadula Model and combined it with the Biba model to create a model [636] that conformed more accurately to the requirements of a commercial policy. For clarity, we consider the Bell-LaPadula aspects of Lipner's model first, and then combine those aspects with Biba's model.

6.3.1. Lipner's Use of the Bell-LaPadula Model

Lipner provides two security levels, in the following order (higher to lower):

- Audit Manager (AM): system audit and management functions are at this level.
- System Low (SL): any process can read information at this level.

He similarly defined five categories:

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

Lipner then assigned users to security levels based on their jobs. Ordinary users will use production code to modify production data; hence, their clearance is (SL, { PC, PD }). Application developers need access to tools for developing their programs, and to a category for the programs that are being developed (the categories should be separate). Hence, application programmers have (SL, { D, T }) clearance. System programmers develop system programs and, like application programmers, use tools to do so; hence, system programmers should have clearance (SL, { SD, T }). System managers and auditors need system high clearance, because they must be able to access all logs; their clearance is (AM, { D, PC, PD, SD, T }). Finally, the system controllers must have the ability to downgrade code once it is certified for production, so other entities cannot write to it; thus, the clearance for this type of user is (SL, { D, PC, PD, SD, T }) with the ability to downgrade programs. These security levels are summarized as follows.

Users	Clearance

Users	Clearance
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

The system objects are assigned to security levels based on who should access them. Objects that might be altered have two categories: that of the data itself and that of the program that may alter it. For example, an ordinary user needs to execute production code; hence, that user must be able to read production code. Placing production code in the level (SL, { PC }) allows such access by the simple security property of the Bell-LaPadula Model. Because an ordinary user needs to alter production data, the *-property dictates that production data be in (SL, { PC, PD }). Similar reasoning supplies the following:

Objects	Class
Development code/test data	(SL, { D, T })
Production code	(SL, { PC })
Production data	(SL, { PC, PD })
Software tools	(SL, { T })
System programs	(SL, Ø)
System programs in modification	(SL, { SD, T })
System and application logs	(AM, { **appropriate categories** })

All logs are append-only. By the *-property, their classes must dominate those of the subjects that write to them. Hence, each log will have its own categories, but the simplest way to prevent their being compromised is to put them at a higher security level.

We now examine this model in light of the requirements in Section 6.1.

1. Because users do not have execute access to category T, they cannot write their own programs, so requirement 1 is met.
2. Application programmers and system programmers do not have read or write access to category PD, and hence cannot access production data. If they do require production data to test their programs, the data must be downgraded from PD to D, and cannot be upgraded (because the model has no upgrade privilege). The downgrading requires intervention of system control users, which is a special process within the meaning of requirement 2. Thus, requirement 2 is satisfied.

3. The process of installing a program requires the downgrade privilege (specifically, changing the category of the program from D to PC), which belongs only to the system control users; hence, only those users can install applications or system programs. The use of the downgrade privilege satisfies requirement 3's need for a special process.
4. The control part of requirement 4 is met by allowing only system control users to have the downgrade privilege; the auditing part is met by requiring all downgrading to be logged.
5. Finally, the placement of system management and audit users in AM ensures that they have access both to the system state and to system logs, so the model meets requirement 5.

Thus, the model meets all requirements. However, it allows little flexibility in special-purpose software. For example, a program for repairing an inconsistent or erroneous production database cannot be application-level software. To remedy these problems, Lipner integrates his model with Biba's model.

6.3.2. Lipner's Full Model

Augment the security classifications with three integrity classifications (highest to lowest):

- System Program (ISP): the classifications for system programs
- Operational (IO): the classifications for production programs and development software
- System Low (ISL): the classifications at which users log in

Two integrity categories distinguish between production and development software and data:

- Development (ID): development entities
- Production (IP): production entities

The security category T (tools) allowed application developers and system programmers to use the same programs without being able to alter those programs. The new integrity categories now distinguish between development and production, so they serve the purpose of the security tools category, which is eliminated from the model. We can also collapse production code and production data into a single category. This gives us the following security categories:

- Production (SP): production code and data
- Development (SD): same as (previous) security category Development (D)
- System Development (SSD): same as (previous) security category System Development (SD)

The security clearances of all classes of users remain equivalent to those of the model without integrity levels and categories. The integrity classes are chosen to allow modification of data and programs as appropriate. For example, ordinary users should be able to modify production data, so users of that class must have write access to integrity category IP. The following listing shows the integrity classes and categories of the classes of users:

Users	Security clearance	Integrity clearance
Ordinary users	(SL, { SP })	(ISL, { IP })
Application developers	(SL, { SD })	(ISL, { ID })
System programmers	(SL, { SSD })	(ISL, { ID })
System controllers	(SL, { SP, SD }) and downgrade privilege	(SP, { IP, ID })
System managers and auditors	(AM, { SP, SD, SSD})	(ISL, { IP, ID })
Repair	(SL, { SP })	(ISL, { IP })

The final step is to select integrity classes for objects. Consider the objects Production Code and Production Data. Ordinary users must be able to write the latter but not the former. By placing Production Data in integrity class (ISL, { IP }) and Production Code in class (IO, { IP }), an ordinary user cannot alter production code but can alter production data. Similar analysis leads to the following:

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, Ø)	(IO, { ID })
System programs	(SL, Ø)	(ISP, { IP, ID })
System programs in modification	(SL, { SSD, })	(ISL, { ID })
System and application logs	(AM, { **appropriate categories** })	(ISL, Ø)
Repair	(SL, { SP })	(ISL, { IP })

The repair class of users has the same integrity and security clearance as that of production data, and so can read and write that data. It can also read production code (same security classification and (IO, { IP }) **dom** (ISL, { IP })), system programs ((SL, { SP }) **dom** (SL, Ø) and (ISP, { IP, ID }) **dom** (ISL, { IP })), and repair objects (same security classes and same integrity classes); it can write, but not read, the system and application logs (as (AM, { SP }) **dom** (SL, { SP }) and (ISL, { IP }) **dom** (ISL, Ø)). It cannot access development code/test data (since the security categories are disjoint), system programs in modification (since the integrity categories are disjoint), or software tools (again, since the integrity categories are disjoint). Thus, the repair function works as needed.

The reader should verify that this model meets Lipner's requirements for commercial models.

6.3.3. Comparison with Biba

Lipner's model demonstrates that the Bell-LaPadula Model can meet many commercial requirements, even though it was designed for a very different purpose. The resiliency of that model is part of its attractiveness; however, fundamentally, the Bell-LaPadula Model restricts the flow of information. Lipner notes this, suggesting that combining his model with Biba's may be the most effective.