The Bell and La Padula Security Model

John Rushby
Computer Science Laboratory
SRI International
Menlo Park CA 94025 USA

Abstract

A precise description is given of the Bell and La Padula security model using modern notation. The development faithfully follows that of the original presentation [1, 2]. The paper is intended to provide a basis for more exact, formal, and scientific discussion of the model than has been the case heretofore.

1 Introduction

In order to study the problem of computer security with any rigor, it is first necessary to have a precise formal description of what is meant by "security" in the context of a computer system. Such *security models* provide a foundation for the development of techniques for specifying and verifying the security of computer systems.

Two of the earliest security models, and by far the most influential, were those developed at the Mitre Corporation by Bell and La Padula [2] and at SRI International by Feiertag, Levitt and Robinson [5]. Each of these models has served as a basis for verifying the security properties of real systems and the SRI model provided a foundation for the development of an automatic tool for checking the security of specifications written in the SPECIAL specification language [4].

The security model developed by Bell and La Padula appears considerably different from that due to Feiertag, Levitt and Robinson; it is an interesting and important question, therefore, to enquire to what extent the differences are merely notational, and to what extent they address fundamentally different aspects of the security modeling problem.

Comparison of the two models is greatly complicated by the fact that they use different mathematical formalisms: Bell and La Padula use the language of General Systems Theory introduced by Mesarović [10], while Feiertag, Levitt and Robinson use a more conventional finite-state machine model [7]. In order to facilitate comparison of the substance of the two models, it seems necessary that both should first be expressed within the same formalism. Since most readers find the language

of General Systems Theory to be rather more complex and much less familiar than elementary automata theory, I have chosen to reformulate the Bell and La Padula model within the SRI formalism. The purpose of the present report is to document this reformulation. It is my intention to present the work of Bell and La Padula as accurately as possible, subject to the constraints of presenting their work within a different notational framework than that originally employed. I have tried to avoid "improving" Bell and La Padula's work; where I have been unable to suppress the desire to do so, I have interpolated my own observations in footnotes or in sections labeled as "commentary."

The technical development presented here is closely based upon that given in the Appendix to Reference [2]—which I believe to be the definitive statement of the Bell and La Padula model. I have also used material from a series of earlier reports [1] where it seems illuminating to do so.

2 Preliminaries

The main difference between the formulation of security given by Bell and La Padula [2] and that to be developed here concerns the choice of a formal model for the notion of "computer system." The model used by Feiertag, Levitt and Robinson, and which is the one to be used here also, is a conventional finite-state automaton model. That is to say, a system is regarded as comprising sets of users (or subjects), states, commands (i.e., inputs), and outputs. "Interpreter functions" supply the identity of the next state, and the output produced, when a given user invokes a given command in a particular state. This system model implicitly determines an association between the sequences of commands applied to the machine and the sequences of outputs produced as a result. On the other hand, General Systems Theory, as used by The Bell and La Padula, *directly* identifies the notion of system with the association between input and output sequences; interpreter functions are not present.

The Bell and La Padula model also differs from that of Feiertag, Levitt and Robinson by including notions of a hierarchical object structure and of "discretionary" security. The first of these is rather specific to the Multics system which was the intended application of the model and has been omitted from my treatment. Discretionary security (in the form of an access control matrix) can easily be added to any security model and is omitted from the present treatment.

3 Definitions

We use a slightly embellished Mealy-type automaton as our model for computer systems. That is, a *system* (or *machine*) M is composed of

- a set S of *states*, with an *initial state* $s_0 \in S$,
- a set U of *users* (or *subjects* in security parlance),[1]
- a set C of *commands* (or *operations*), and
- a set O of *outputs*,

together with the functions *next* and *out*:

- next : $S \times U \times C \to S$,
- out : $S \times U \times C \to O$.

If the system is in state s when the user u invokes operation c, then the value $out(s, u, c)$ will be output and the next state of the system will be $next(s, u, c)$. Pairs of the form $(u, c) \in U \times C$ are called *actions*.

We derive a function *next*$\star$:

- next$\star$: $S \times (U \times C)^* \to S$

(the natural extension of *next* to *sequences* of actions) by the equations

$$\begin{aligned} \text{next}\star(s, \Lambda) &= s, \text{and} \\ \text{next}\star(s, \alpha \circ (u, c)) &= \text{next}(\text{next}\star(s, \alpha), u, c), \end{aligned}$$

where Λ denotes the empty string and $\circ$ denotes string concatenation.

The intuition here is that the machine starts off in the initial state s_0 and is presented with a sequence $\alpha \in (U \times C)^*$ of actions. This causes the machine to progress through a sequence of states, eventually reaching the state $next\star(s_0, \alpha)$.

The system state is not directly visible to the outside world; all that can be observed are the outputs returned in response to the actions presented as inputs. Although the system state is not directly visible or accessible, it records all the information necessary to determine the subsequent behavior of the system. The SRI security model [4–6,12] defines security directly in terms of the relationship between input actions and outputs (basically it states that input actions from highly classified users must have no effect on the outputs subsequently returned to users of lower classification). From this definition it is possible to elaborate the model to address the internal interactions between actions and the system state. Bell and La Padula's model, on the other hand, identifies security with these internal interactions right from the start. In order to do this, the internal structure of the system state must be modeled in more detail and in this regard the notion of an "object" is fundamental

[1] The distinction between users and subjects is irrelevant to the construction of the model—both terms are simply names for members of the abstract set U. The distinction is important, however, when we consider *interpretations* of the model.

to the Bell and La Padula model. Basically, the system state is regarded as a record of the "values" currently "stored" within the objects of the system, together with a record of the type of "access" that each user is currently permitted to each object. We say that the association between object names and values constitutes the *value state* of the system, while the record of the types of access permitted between each subject and object constitutes the *protection state*.

The partitioning of the system state into a value component and a protection component is mirrored by the partitioning of the command set C into *operations*, which can change the value state but not the protection state, and *rules* which can change the protection state but not the value state. (It is, of course, possible to consider commands which change *both* the value and the protection state, but we will not do so.)

The two fundamental types of "access" that can occur between subjects and objects are termed *observation* and *alteration*. Bell and La Padula define the first as the "extraction of information" from an object, and the second as the "insertion of information" into an object. No formal definition of these concepts is provided by Bell and La Padula; interpretation of the model depends on the reader's intuitive understanding of the names used to describe them.

> The distinction between the value and protection state components of the general system state is not made by Bell and La Padula, nor is that between operations and rules. Their exposition concentrates solely on the protection state of the system—they use the simple term "state" where I use "protection state"; they use the term "rule" in the same sense as here, but do not admit the dual notion of "operation."

Based on these two primitive types of access, four more elaborate ones can be constructed. These are known as $w, r, a,$ and e access, respectively:

w: **write** access permits both observation and alteration,
r : **read** access permits observation but not alteration,
a : **append** access permits alteration, but not observation, and
e : **execute** access permits neither observation nor alteration.

In order to model formally this internal structure of the system state we introduce

- a set N of object *names*,
- a set V of object *values*,
- the set $A = \{w, r, a, e\}$ of *access types*,

and also the functions *contents* and *current-access-set*:

- contents : $S \times N \rightarrow V$,
- current-access-set : $S \rightarrow \mathcal{P}(U \times N \times A)$

(where $\mathcal{P}$ denotes *powerset*) with the interpretation that *contents*(s, n) returns the value of object n in state s, while *current-access-set*(s) returns the set of all triples (u, n, x) such that subject u has access type x to object n in state s. Observe that *contents* captures the idea of the *value state*, while *current-access-set* embodies the *protection state* of the system.[2]

> Although execute access permits neither observation nor alteration, the name suggests that execute access is intended to indicate the right to execute an object as a program, and is therefore to be distinguished from the mere absence of both observation and alteration access. However, this is not the case. Bell and La Padula state specifically [2, footnote to page 11]: "this abstract notion of 'execute' access is not what is typically implemented (enforced) by computer hardware since the results of execution reflect the contents and thus constitute 'observation' of the executed element."
>
> There is an unstated assumption concerning the *current-access-set* that is fundamental to the interpretation of their model. This assumption, which I call the *Object Monitor Assumption*, is that the system "hardware" enforces restrictions on user's access to objects in accordance with the permissions recorded in the *current-access-set*. That is to say, the protection state controls access to the value state. Since *rules* do not access the value state, the Object Monitor Assumption is embodied in the following three constraints on the behavior of *operations*. These constraints (which are easily formalized) can also be regarded as supplying interpretation to the informal notions of "observation" and "alteration" rights.
>
> 1. The output of an operation may only depend on the values of objects to which the user who executes the operation has observation rights.
> 2. An operation may only change the values of objects to which the user who executes the operation has alteration rights.
> 3. The new values assigned to objects as a result of executing an operation may only depend on the values of objects to which the user who executes the operation has observation rights.

[2] Actually, *current-access-set* embodies only part of the protection state of the system—additional components are introduced later in order to record subject clearances and object classifications.

Most of the definitions that follow hinge on the distinction between observation and alteration access—whereas the *current-access-set* is expressed in terms of *w, r, a,* and *e* access. Consequently, it is convenient to define functions *observe* and *alter* that extract observation and alteration access from the derived forms. (Bell and La Padula did not adopt this approach, but stated their definitions in terms of the basic *current-access-set* function.) Thus, we introduce functions *alter*, and *observe*:

- alter : $S \to \mathcal{P}(U \times N)$, and
- observe : $S \to \mathcal{P}(U \times N)$

with the definitions:

$$\begin{aligned} \text{observe}(s) &\stackrel{\text{def}}{=} \{(u,n) \mid (u,n,w) \text{ or } (u,n,r) \in \text{current-access-set}(s)\}, \textit{ and} \\ \text{alter}(s) &\stackrel{\text{def}}{=} \{(u,n) \mid (u,n,w) \text{ or } (u,n,a) \in \text{current-access-set}(s)\}. \end{aligned}$$

That is, *observe*(s) returns the set of all subject-object pairs (u,n) for which subject u has *observation* rights to object n in state s, while *alter*(s) returns the set of all pairs for which subject u has *alteration* rights to object n in state s.

The final step in the basic development is to assign a notion of "clearance" to users and "classification" to objects. Both clearances and classifications are modeled by a partially ordered set of "security levels."[3] Bell and La Padula distinguish a user's *clearance* from the security level at which he currently chooses to operate. Whereas a user's clearance is fixed, his current level is recorded in the system state. Similarly, the classifications of objects are also recorded in the system state.

These ideas can be formalized in terms of

- a set L of *security levels*, partially ordered by a relation $\geq$ (pronounced "dominates"),

and functions

- clearance : $U \to L$,
- current-level : $S \times U \to L$, and
- classification : $S \times N \to L$.

The intended interpretation is that $l_1 \geq l_2$ if and only if users with clearance l_1 are allowed access to information with classification l_2. The clearance of user u is denoted by *clearance*(u), while *current-level*(s,u) denotes his current level in state

[3] This is sufficient to encompass both the standard hierarchy of military classifications (UNCLASSIFIED, CONFIDENTIAL, SECRET, TOP SECRET) as well as "need-to-know" compartments (see [3, Chapter 5]).

s, and *classification*(s, n) denotes the classification of object n in state s. Naturally, we require that a user's clearance always dominates his current level:

$$\text{clearance}(u) \geq \text{current-level}(s, u), \forall u \in U \text{ and } \forall s \in S.$$

As noted earlier, the commands of the system are divided into those that change the protection state (now enlarged to contain the functions *current-level* and *classification* as well as *current-access-set*) but not the value state (which commands we call *rules*), and those that change the value state but not the protection state (which commands we call *operations*). It proves convenient to further subdivide the set of rules into those that change the current security levels of subjects or the classifications of objects, and those that do not (which latter group we call *tranquil* rules). [4]

- An *operation* is a command $op \in C$ such that, $\forall s \in S,\ \forall u, v \in U,$ and $\forall n \in N$:

$$\begin{aligned} \text{current-access-set}(\text{next}(s, u, op)) &= \text{current-access-set}(s), \\ \text{current-level}(\text{next}(s, u, op), v) &= \text{current-level}(s, v), \end{aligned}$$

  and

$$\text{classification}(\text{next}(s, u, op), n) = \text{classification}(s, n).$$

- A *rule* is a command $r \in C$ such that, $\forall s \in S,\ \forall u, v \in U,$ and $\forall n \in N$:

$$\text{contents}(\text{next}(s, u, r), n) = \text{contents}(s, n).$$

- A rule $r \in C$ is *tranquil* if, $\forall s \in S,\ \forall u, v \in U,$ and $\forall n \in N$,

$$\begin{aligned} \text{current-level}(\text{next}(s, u, r), v) &= \text{current-level}(s, v), \text{ and} \\ \text{classification}(\text{next}(s, u, r), n) &= \text{classification}(s, n). \end{aligned}$$

- A system is *simple* if all of its commands are either operations or rules.[5]

We are now ready to state Bell and La Padula's definition of security. By analogy with the "pen and paper" world, it is plainly necessary that users should only be able to observe the values of objects whose classifications are dominated by their own clearance. Thus, if user u has r or w access to object n in state s, we require that the maximum level of u must dominate the classification of n in state s.[6] This is called the *simple security property* (or *ss-property* for short).

[4] Introduction of the term "tranquility" in the sense employed here is generally attributed to Bell and La Padula, but I can find no mention of it in reference [2], where those results that require tranquility cite the formal statement of the property but do not give it a name.

[5] It is implicit in Bell and La Padula's model that only simple systems are considered.

[6] It might seem more natural to require that the *current* level of u should dominate the classification of n. This additional restriction is indeed imposed (but only for *untrusted subjects*) as part of the $\star$-property.

Definition 1 (Simple Security Property) A state $s \in S$ satisfies the simple security property if $\forall u \in U$, and $\forall n \in N$:

$$(u, n) \in \text{observe}(s) \quad \supset \quad \text{clearance}(u) \geq \text{classification}(s, n).$$

A rule r is *ss-property-preserving* if $next(s, u, r)$ satisfies the ss-property whenever s does. □

Consideration of the possibility that untrustworthy subjects might circumvent the intent, though not the statement, of the simple security property by copying highly classified material into objects of lower classification then led Bell and La Padula to formulate their famous $\star$-*property* (pronounced "star-property"). The motivation for the $\star$-property was rather well expressed by Bell and La Padula as follows:

> "The expected interpretation of the model anticipates protection of information containers rather than of the information itself. Hence a malicious program (an interpretation of a subject) might pass classified information along by putting it into an information container labeled at a lower level than the information itself." [2, p16]

In formulating the $\star$-property, Bell and La Padula first divided the subjects of the system into those that may be *trusted* to place information only in appropriately labeled containers, and those that may not be so trusted. The $\star$-property then prohibits untrusted subjects from "writing down"—that is to say it prohibits them from acquiring alteration rights to objects classified below their own current level. It also prohibits untrusted subjects from reading above their *current* level. (Recall that the ss-property prohibits *all* subjects from reading above their *maximum* level or clearance.)

Definition 2 ($\star$-property) Let $T \subseteq U$ denote the set of *trusted subjects*. A state $s \in S$ satisfies the $\star$-property if, for all *untrusted* subjects $u \in U \backslash T$ (we use $\backslash$ to denote set difference) and objects $n \in N$:

$$\begin{aligned} (u, n) \in \text{alter}(s) \quad &\supset \quad \text{classification}(s, n) \geq \text{current-level}(s, u), \text{ and} \\ (u, n) \in \text{observe}(s) \quad &\supset \quad \text{current-level}(s, u) \geq \text{classification}(s, n). \end{aligned}$$

A rule r is $\star$-*property-preserving* if $next(s, u, r)$ satisfies the $\star$-property whenever s does. □

Note that it follows from these definitions that:

$$\begin{aligned} (u, n, a) \in \text{current-access-set}(s) \quad &\supset \quad \text{classification}(s, n) \geq \text{current-level}(s, u), \\ (u, n, r) \in \text{current-access-set}(s) \quad &\supset \quad \text{current-level}(s, u) \geq \text{classification}(s, n), \end{aligned}$$

and

$$(u, n, w) \in \text{current-access-set}(s) \quad \supset \quad \text{classification}(s, n) = \text{current-level}(s, u).$$

Also, as a simple consequence of the transitivity of $\geq$, if a state s satisfies the $\star$-property and u is an untrusted subject with alteration rights to object n_1 and observation rights to object n_2 (in state s), then

$$\text{classification}(s, n_1) \geq \text{classification}(s, n_2).$$

The original formulation of the $\star$-property was somewhat different than that given above in that it did not employ the notion of a subject's *current-level*. The formulation of the $\star$-property given in [1, Volume II] is, $\forall u \in T \backslash U$, and $\forall m, n \in N$:

$$\begin{aligned}(u, m) \in \text{observe}(s) \ \wedge \ (u, n) \in \text{alter}(s) \\ \supset \text{classification}(s, n) \geq \text{classification}(s, m).\end{aligned}$$

That is, the classification of any object to which a subject has alteration rights must dominate that of any object to which it has observation rights. In [1, Volume III] is was noted that if a state satisfies the $\star$-property as formulated above, and if an untrusted subject u has write access to two objects simultaneously, then both those objects must have the *same* classification l, say. Furthermore, the classification of any object to which the subject has append access must dominate l, while l itself must dominate the classification of any object to which the subject has read access. Thus, the classification l is a pivot to which the classifications of all other accessible objects must relate. Accordingly, in [1, Volume III], Bell and La Padula distinguished this classification as the "current-level" of the subject concerned and provided the reformulation of the $\star$-property given in Definition 2. They reported that this formulation of the $\star$-property considerably simplified a number of the tests that need to be performed in any implementation of the model.

Bell and La Padula identify "security" with the conjunction of the ss- and $\star$-properties.

Definition 3 (Security)

- A state is *secure* if it satisfies both the simple security property and the $\star$-property.
- A rule r is *security-preserving* if $next(s, u, r)$ is secure whenever s is.

- We say that a state s is *reachable* if $s = next\star(s_0, \alpha)$ for some action sequence $\alpha \in (U \times C)^*$.
- A system satisfies the simple security property if every reachable state satisfies the simple security property.
- A system satisfies the $\star$-property if every reachable state satisfies the $\star$-property.
- A system is secure if every reachable state is secure.

4 Results

In this section, we establish conditions on individual state transitions that are sufficient to ensure the security of the system overall. I include all the results in the Appendix to [2], except those relating to discretionary security. In contrast to Bell and La Padula, I provide only informal outlines for the proofs of these results: when presented in the formalism used here, the results are so obvious that more elaborate proofs seem superfluous.

The first result establishes the conditions under which a system satisfies the ss-property.

Theorem 1 (Theorem A1 of [2]) A system satisfies the ss-property if and only if:

1. The initial state s_0 satisfies the ss-property, and
2. $\forall s, t \in S,\ \forall u, v \in U,\ \forall n \in N,$ and $\forall c \in C,$

   (a) $t = \text{next}(s, v, c)\ \wedge\ (u, n) \in \text{observe}(t) \backslash \text{observe}(s)$
   $\supset \text{clearance}(u) \geq \text{classification}(t, n)$, and

   (b) $t = \text{next}(s, v, c)\ \wedge\ (u, n) \in \text{observe}(s)$
   $\wedge\ \text{clearance}(u) \not\geq \text{classification}(t, n)$
   $\supset (u, n) \notin obs(t)$.

Proof: The proof is an elementary induction on the number of actions required to reach each (reachable) state. The first hypothesis in the statement of the theorem provides the basis of the induction; the second provides the inductive step. The first part of the second hypothesis asserts that no observation right is ever *added* to the protection state if it would cause the new state to violate the ss-property, while the second part asserts that no observation right *persists* from one state to the next if it would cause the new state to violate the ss-property. □

Next we establish the conditions under which a system satisfies the $\star$-property.

Theorem 2 (Theorem A2 of [2]) A system satisfies the $\star$-property if and only if:

1. The initial state s_0 satisfies the $\star$-property, and
2. $\forall s, t \in S,\ \forall v \in U,\ \forall u \in U \backslash T,\ \forall n \in N,$ and $\forall c \in C,$

   (a) $t = \text{next}(s, v, c) \ \wedge\ (u, n) \in \text{observe}(t) \backslash \text{observe}(s)$
   $\supset \text{current-level}(t, u) \geq \text{classification}(t, n),$

   (b) $t = \text{next}(s, v, c) \ \wedge\ (u, n) \in \text{alter}(t) \backslash \text{alter}(s)$
   $\supset \text{classification}(t, n) \geq \text{current-level}(t, u),$

   (c) $t = \text{next}(s, v, c) \ \wedge\ (u, n) \in \text{observe}(s)$
   $\wedge\ \text{current-level}(t, u) \not\geq \text{classification}(t, n)$
   $\supset (u, n) \notin \text{observe}(t),$ and

   (d) $t = \text{next}(s, v, c) \ \wedge\ (u, n) \in \text{alter}(s)$
   $\wedge\ \text{classification}(t, n) \geq \text{current-level}(t, u)$
   $\supset (u, n) \notin \text{alter}(t).$

Proof: Again, the proof is an elementary induction on the number of actions required to reach each (reachable) state and is very similar to that of the previous theorem. The first hypothesis provides the basis for the induction while the second provides the inductive step. The first two parts of the second hypothesis assert that no observation or alteration rights are ever added to the protection state if they would cause the new state to violate the $\star$-property; the final two parts of that hypothesis ensure that no rights *persist* form one state to the next if they would cause the new state to violate the $\star$-property. □

Combining these two results yields:

Corollary 3 (Basic Security Theorem—Corollary A1 of [2])
A system is secure if and only if it satisfies the hypotheses of both the previous theorems. □

> The hypotheses to the Basic Security Theorem are so unhelpful that the theorem cannot serve as a basis for proving the security of real systems. Indeed, no subsequent use of the theorem was made by Bell and La Padula. Their motive in stating the result was merely to establish the "inductive" nature of their definition of security in that
>
> > "it shows that the preservation of security from one state to the next guarantees total system security" [2, p20].
>
> Bell and La Padula contrasted the inductive nature of security with the more complex problem of deadlock-avoidance in certain resource-sharing systems and observed that the Basic Security Theorem establishes

> "the relative simplicity of maintaining security: the minimum check that the proposed new state is 'secure' is both necessary and sufficient for full maintenance of security" [2, p21].

Despite the modest claims made by Bell and La Padula for the Basic Security Theorem, the result is sometimes advanced as a substantial argument in favor of the proposition that the Bell and La Padula security model captures the "essence" of security and is superior to other models [8]. McLean [9] attacks this argument by establishing an essentially identical theorem for a model that clearly violates any reasonable notion of "security."[7] In fact, it should be clear that if Φ is *any* effectively decidable property of the system state, then an analogue to the Basic Security Theorem can be constructed for that Φ. In other words, and as McLean observed, the Basic Security Theorem is a property of the finite-state system model employed (in that states can be indexed to support proof by induction), rather than of the particular definition given for security.

Clearly we have:

Theorem 4 (Theorem A4 of [2]) If s_0 satisfies the ss-property and all the rules of the simple system M are ss-property preserving, then M satisfies the ss-property.

Proof: Immediate from the definitions of *simple system*, *rule* and *ss-property-preserving.* □

And

Theorem 5 (Theorem A5 of [2]) If s_0 satisfies the $\star$-property and all the rules of the simple system M are $\star$-property preserving, then M satisfies the $\star$-property.

Proof: Immediate from the definitions of *simple system*, *rule* and *$\star$-property-preserving.* □

And hence

Corollary 6 (Corollary A2 of [2]) If s_0 is a secure state and all the rules of the system M are security-preserving, then M is a secure system. □

Also

[7] Basically, McLean turns the $\star$-property around (to yield what he calls the "%-property"), so that subjects may (only) transfer information from higher to lower classification levels.

Theorem 7 (Theorem A7 of [2]) Let r be a tranquil rule such that, $\forall s \in S$, and $\forall u \in U$:

$$\text{observe}(\text{next}(s, u, r)) \supset \text{observe}(s).$$

Then r is ss-property-preserving if, $\forall s \in S$, $\forall u, v \in U$, and $\forall n \in N$,

$$\begin{aligned}(v, n) \in \text{observe}(\text{next}(s, u, r))\backslash\text{observe}(s) \\ \supset \text{clearance}(v) \geq \text{classification}(s, n).\end{aligned}$$

Proof: The result is obvious: it simply states that if a tranquil rule (i.e., one that does not change the security levels assigned to subjects and objects) merely adds new observation rights to those already present, and if each of those newly added rights satisfies the conditions of the ss-property, then the rule is ss-property-preserving. □

Theorem 8 (Theorem A8 of [2]) Let r be a tranquil rule such that, $\forall s \in S$, and $\forall u \in U$:

$$\begin{aligned}\text{alter}(\text{next}(s, u, r)) &\supset \text{alter}(s), \text{ and} \\ \text{observe}(\text{next}(s, u, r)) &\supset \text{observe}(s)\end{aligned}$$

Then r is $\star$-property-preserving if, $\forall s \in S$, $\forall u \in U$, $\forall v \in U\backslash T$, and $\forall n \in N$:

$$\begin{aligned}(v, n) \in \text{alter}(\text{next}(s, u, r))\backslash\text{alter}(s) \\ \supset \text{classification}(s, n) \geq \text{current-level}(s, v),\end{aligned}$$

and

$$\begin{aligned}(v, n) \in \text{observe}(\text{next}(s, u, r))\backslash\text{observe}(s) \\ \supset \text{current-level}(s, v) \geq \text{classification}(s, n).\end{aligned}$$

Proof: Again the result is obvious: it simply states that if a tranquil rule (i.e., one that does not change the security levels assigned to subjects and objects) merely adds new observation and alteration rights to those already present, and if each of those newly added rights satisfies the conditions of the $\star$-property, then the rule is $\star$-property-preserving. □

Corollary 9 (Corollary A3 of [2]) Let r be a tranquil rule such that, $\forall s \in S$, and $\forall u \in U$:

$$\begin{aligned}\text{alter}(\text{next}(s, u, r)) &\supset \text{alter}(s), \text{ and} \\ \text{observe}(\text{next}(s, u, r)) &\supset \text{observe}(s).\end{aligned}$$

Then r is security-preserving if the conditions of the previous two theorems are satisfied. □

Theorem 10 (Theorem A10 of [2]) Let r be a tranquil rule such that, $\forall s \in S$, and $u \in U$:

$$\text{observe}(\text{next}(s, u, r)) \subseteq \text{observe}(s).$$

Then r is ss-property-preserving. If, in addition,

$$\text{alter}(\text{next}(s, u, r)) \subseteq \text{alter}(s),$$

then r is $\star$-property-preserving, ss-property-preserving, and security-preserving.

Proof: Yet again, this is an obvious result: it simply states that if a tranquil rule (i.e., one that does not change the security levels assigned to subjects and objects) merely *removes* observation and/or alteration rights from those already present, then the rule is ss-property, $\star$-property, or security-preserving as appropriate. □

5 Applications

Bell and La Padula demonstrated the application of their security model by using the results of the previous section to establish the security of a representative class of 11 rules. These rules were chosen to model those found in the Multics system. I will outline a few examples from this application (omitting rules 6 through 9 of [2], which are concerned with discretionary security and the Multics object hierarchy).

5.1 Get-Read (rule 1 of [2])

A subject u may call the rule *get-read(n)* in order to acquire read access to the object n. The rule checks that the following conditions are satisfied.

1. $\text{clearance}(u) \geq \text{classification}(s, n)$.
2. If u is not a trusted subject (i.e., $u \in U \backslash T$), then

$$\text{current-level}(s, u) \geq \text{classification}(s, n).$$

If both these conditions are satisfied, the rule modifies the protection state by setting

$$\text{current-access-set}(s') = \text{current-access-set}(s) \cup \{(u, n, r)\},$$

where s' denotes the new system state following execution of the rule. Otherwise, the system state is not modified.

The security of get-read follows directly from Corollary 9.

5.1.1 Get-Append, Get-Execute, Get-Write (rules 2 to 4 of [2])

These are analogous to get-read.

5.2 Release-Read (rule 5 of [2])

A subject u may call the rule *release-read(n)* in order to release its read access right to the object n. No checks are made by the rule, which simply modifies the protection state by setting

$$\text{current-access-set}(s') = \text{current-access-set}(s) \backslash \{(u, n, r)\},$$

where s' denotes the new system state following execution of the rule.

The security of release-read follows directly from Theorem 10.

5.2.1 Release-Execute, Release-Append, Release-Write (rule 5 of [2])

These are analogous to release-read.

5.3 Change-Subject-Current-Security-Level (rule 10 of [2])

A subject u may call *Change-Subject-Current-Security-Level(l)* in order to request that its current-level be changed to l. The rule checks that the following conditions are satisfied.

1. $\text{clearance}(u) \geq l$ (i.e., a subject's *current-level* may not exceed its *clearance*).
2. If u is an untrusted subject (i.e., $u \in U \backslash T$) then assigning l as the current level of u must not cause the resulting state to violate the $\star$-property—i.e., $\forall n \in N$:

$$\begin{aligned} (u, n) \in \text{alter}(s) &\supset \text{classification}(s, n) \geq l, \text{ and} \\ (u, n) \in \text{observe}(s) &\supset l \geq \text{classification}(s, n). \end{aligned}$$

If both these conditions are satisfied, the rule modifies the system state by setting *current-level*$(s', u) = l$, where s' denotes the new system state following execution of the rule. Otherwise, the system state is not modified.

The ss-property is independent of a subject's *current-level*; hence this rule is ss-property preserving. By its very construction, the rule is $\star$-property preserving. Hence it is security preserving.

> Although this rule preserves security as it is defined here (i.e., as the conjunction of the ss- and $\star$-properties), it still provides a "leakage channel" by which an untrustworthy subject can indirectly transfer information from a highly classified object to one of lower classification. This leakage channel is described by Millen and Cerniglia [11] who attribute its discovery to P.S. Tasker of the Mitre Corporation. Its method of operation is as follows:

1. An untrustworthy subject (a "Trojan Horse") with a high current-level extracts sensitive information from an object to which it has (legitimate) read access.
2. It then encodes this information by selecting to which members of a set of lowly classified objects it acquires (legitimate) read access.
3. The subject then releases read access to the highly classified file and reduces its current level to that of the lowly classified objects.
4. It then observes to which of these objects it has read access and thereby recovers the highly classified information encoded thereby.
5. Finally it writes the recovered information into an object at its current (low) level—thereby making it available to subjects of low clearance.

5.4 Change-Object-Security-Level (rule 11 of [2])

A subject u may call *Change-Object-Security-Level(n, l)* in order to request that the classification of object n be changed to l. The rule checks that the following conditions are satisfied.

1. current-level$(s, u) \geq$ classification(s, n) (i.e., no subject may change the classification of an object which is currently classified above its own level).
2. If u is an untrusted subject (i.e., $u \in U \backslash T$), then

   $$\text{current-level}(s, u) \geq l \text{ and } l \geq \text{classification}(s, n),$$

   (i.e., untrusted subjects may not "downgrade" the classification of an object).
3. $\forall v \in U, (v, n) \in \text{observe}(s) \supset \text{current-level}(s, v) \geq l$ (i.e., if any subject has observation rights to the object n, then the current level of that subject must dominate the new classification of n).
4. Assigning l as the classification of n must not cause the resulting state to violate the $\star$-property.

If these conditions are satisfied, the rule modifies the system state by setting $\mathit{classification}(s', n) = l$, where s' denotes the new system state following execution of the rule. Otherwise, the system state is not modified.

The third condition is (more than) sufficient to ensure that this rule is ss-property preserving. The fourth condition explicitly guarantees that it is $\star$-property preserving—hence the rule is security preserving.

Only the third and fourth conditions are necessary in order to ensure that this rule satisfies Bell and La Padula's definition of security—yet the second condition is utterly essential to the preservation of any "real" security (the first condition is more of an "integrity" constraint).

Like the previous rule, this one also provides a "leakage channel" which allows information to flow from a highly classified object to a subject with a lower clearance. The scenario is as follows.

1. An untrustworthy subject (a "Trojan Horse") with a high current-level extracts sensitive information from an object to which it has (legitimate) read access.
2. It then encodes this information by selecting which members of a set of lowly classified objects it will "upgrade" (i.e., raise in security classification) from level l_1 to a slightly higher level l_2.
3. A confederate subject with a low l_2 clearance discovers which objects have changed in classification from l_1 to l_2—thereby recovering the highly classified information which the Trojan Horse has encoded in this way.

Another potential security flaw in this rule is described by Taylor [13]. Consider a trusted subject copying information between two objects of the same classification—l_1, say. It is possible for a second subject to raise the classification of the source object to a level l_2, say, which is above that of the destination object. This second subject could then copy some l_2 level material into the source file, thereby turning the trusted subject into an unwitting channel for unsecure information flow from level l_2 to level l_1. [8] Although a trusted subject is trusted not to abuse its potential to cause unsecure information flow, it is eminently possible that those who design and validate a program which partakes of the role of a trusted subject might overlook the possibility that the classifications of the objects it manipulates can change beneath it.

This second channel (but not the first) can be prevented if the rule is modified to refuse changes to the classifications of *active* objects—those are objects to which some subject has current access. More formally, an object $n \in N$ is *active* in state $s \in S$ if there exists a subject $u \in U$ and an access mode $x \in A$ such that $(u, n, x) \in$ *current-access-set*(s).

[8] In order to satisfy the conditions in the rule, both subjects must have current levels which dominate both levels l_1 to l_2.

References

[1] D. E. Bell and L. J. La Padula. Secure computer systems: Vol. I—mathematical foundations, Vol. II—a mathematical model, Vol. III—a refinement of the mathematical model. Technical Report MTR-2547 (three volumes), Mitre Corporation, Bedford, MA, March–December 1973.

[2] D. E. Bell and L. J. La Padula. Secure computer system: Unified exposition and Multics interpretation. Technical Report ESD-TR-75-306, Mitre Corporation, Bedford, MA, March 1976.

[3] D. E. Denning. *Cryptography and Data Security.* Addison-Wesley, 1982.

[4] R. J. Feiertag. A technique for proving specifications are multilevel secure. Technical Report CSL-109, Computer Science Laboratory, SRI International, Menlo Park, CA, January 1980.

[5] R. J. Feiertag, K. N. Levitt, and L. Robinson. Proving multilevel security of a system design. In *Sixth ACM Symposium on Operating System Principles*, pages 57–65, November 1977.

[6] J. A. Goguen and J. Meseguer. Security policies and security models. In *Proceedings of the Symposium on Security and Privacy*, pages 11–20, Oakland, CA, April 1982. IEEE Computer Society.

[7] J. E. Hopcroft and J. D. Ullman. *Formal Languages and their Relation to Automata.* Addison-Wesley, 1969.

[8] S. B. Lipner (Moderator). Panel session: Bell/La Padula and alternative models of security. In *Proceedings of the Symposium on Security and Privacy*, Oakland, CA, April 1983. IEEE Computer Society.

[9] J. McLean. A comment on the "basic security theorem" of Bell and La Padula. Informal note, Naval Research Laboratory, 1983.

[10] M. D. Mesarović. A mathematical theory of general systems. In G. J. Klir, editor, *Trends in General Systems Theory.* John Wiley and Sons, 1972.

[11] J. K. Millen and C. M. Cerniglia. Computer security models. Working Paper WP25068, Mitre Corporation, Bedford, MA, September 1983.

[12] John Rushby. The security model of Enhanced HDM. In *Proceedings 7th DoD/NBS Computer Security Initiative Conference*, pages 120–136, Gaithersburg, MD, September 1984.

[13] T. Taylor. Comparison paper between the Bell and La Padula model and the SRI model. In *Proceedings of the Symposium on Security and Privacy*, pages 195–202, Oakland, CA, April 1984. IEEE Computer Society.