Some Lessons from Formalizing a Security Model

Carl E. Landwehr

Computer Science and Systems Branch
Naval Research Laboratory
Washington, D.C. 20375-5000

1. Introduction

A significant problem in verifying programs or systems is determining the properties (assertions, invariants, etc.) that are to be verified. At the highest level of specification, these properties are usually defined only informally. For program verification techniques to assure that a program or system of programs maintains these properties, they must first be cast in a formal notation. This task is nontrivial, and although the final product (the formal statement) is visible, the considerations that lead from the informal to the formal statement are rarely described. This paper documents some of those considerations for a particular security model.

At the first Verkshop, a set of assertions was proposed for verification of multilevel secure military message systems. These assertions were stated in English together with a set of definitions, a description of system operation, and a set of assumptions [Land80]. The purpose of that exposition was to provide a definition of security based on the message system application that is comprehensible to users, designers, and implementors of the system. Because the model was stated in English rather than in mathematical notation, it was easier for individuals not trained in mathematics to understand and comment upon. Considerations of the informal model and the message system application led to substantial changes (mostly additions) to the model. These included the generalization of access lists into access sets, the definition of direct and indirect references and the container-clearance-required constraint, the definition of user roles, and the requirement that all output be labeled. Three rapid prototypes based on the informal model have been built [Heit82,Corn84]. Efforts to formalize the model proceeded in parallel with the construction of these prototypes, and the revised informal model and a formalization of it have been published [Land84]; for brevity we will not review the published work here.

2. Keep the Application in Mind

English words can have many meanings. Assertion 3 of the informal model states that "information removed from an object inherits the classification of that object" and "information inserted into an object must not have a classification higher than the classification of that object." In the context of typical message system operations, the interpretation of these statements may seem straightforward: removing a text string from one object and inserting it into another must follow the specified rules. But "information" is an elusive word, and an apparently straightforward interpretation could lead to a formal model that seems to correspond to the informal one yet is not secure.

For example, suppose an operation TH consults the value of a top secret object, locates a confidential object that has the same value and then displays the confidential object on a confidential output device. No physical change to the confidential value occurred, no classification marking was altered, and no data labeled above the level of the output device was displayed on it.

On the other hand, suppose an operation OK scans a user's top secret message file (e.g., his inbox) but displays only messages that are at a security level at or below that of his output device. Like TH, operation OK consults the value of a top secret entity and displays confidential outputs.

To define a formal model that prohibits TH and permits OK is not trivial. Our solution was to define "potential modification" and "contributing factor" and apply these in the definition of "copy-secure" (Definition 6 of the formal model). Roughly, an entity is potentially modified by an operation if there is some system state (e.g., in which some entity values are different) in which invoking this operation would alter the value of this entity. Each entity whose value makes a difference in this respect is a contributing factor to the potential modification. Copy-secure requires that the classification of each entity potentially modified by an operation dominate that of each contributing factor.

In the examples just described, operation TH is prohibited since changing the value of the top secret data would change the value contained by the confidential output device. The operation OK can be permitted, since changes to the top secret data in the inbox would not affect the value contained by the output device.

The definitions of potential modification, contributing factor, and copy secure were all much worried over. The point here is not that this particular formalization successfully defines security for message system, but that in practice it is frequently necessary to consider specific scenarios to determine whether a formalization on one hand permits the operations that a particular secure application must support and on the other prohibits those that are intuitively insecure.

3. Control Operations with Incompletely Specified Semantics

A security model that includes a formal semantics of all operations in the system is impractical. Such a model would change every time an operation is added or modified. On the other hand, it is useful to be able to assure that a system user can invoke only operations for which he is authorized. For example, only particular authorized users ought to be permitted to alter the clearance of a user. It is straightforward to limit operations that change user clearances, however, only because part of the semantics of the operation have been specified, so a post-condition can be written based on that: if a user's clearance in the new state differs from its value in the old state, then the operation must have been invoked by the security officer and (to prohibit clearance changes as unwanted side effects) the operation must have been the set-clearance operation.

Consider now an operation like display-message. For discretionary access controls to have some meaning, it is necessary that a user be able to display a particular message only if he has permission to do so (in addition to having the necessary clearance). In the MMS security model, such permission is reflected by the presence of the appropriate tuple in the access set for the message. But to define a post-condition for each possible operation restricting its invocation to cases where the appropriate entries are in its access set based (as in set-clearance) on the effects of that operation leads directly to specifying the formal semantics of each operation as part of the security model.

The resolution of this issue in the formal MMS security model is contained in Definition 5, access-secure. It requires that if there is any change at all in the system state, then each operand that is an entity must have had the appropriate entry in its access set. While this property seems adequately to represent the permission to invoke an operation, there is still a question of restricting the side effects of an operation. In the set-clearance example above, the requirement that if a clearance was changed, then the set-clearance operation was invoked, prevents any other operation from having the side effect of altering clearances. This approach can't be used to control side effects of operations whose semantics are not defined. It might be argued that this is unimportant: the only issue is that whatever side effects there are obey the security assertions. It seems unsatisfactory, however, to define a security model that permits operations that might, for example, modify entities that do not appear as operands even if those modifications satisfy the security model constraints.

After attempting to express this kind of constraint directly in the formal assertions, we decided it would be simpler to assume the existence of a unique operation that changes a user or entity with respect to a particular function (value, type, classification, etc.). More complex operations (with unspecified semantics) must be able to be expressed in terms of these operations, which are not permitted to have side effects. This prevents an operation that is intended only to alter the access set of a message from, for example, changing its text as well unless the value-changing operation is invoked explicitly. This solution has much in common with the constraint defined by Popek and Farber [Pope78] that "protected objects are modified only by explicit request."

4. Omit Awkward Inessentials

The question of how to handle error messages in the formal model also arises in the context of access-secure. Suppose a user attempts to modify a field of a message he is not permitted to display, but that the proposed modification is identical to the current value of that field. (That is, he attempts to guess the value of the field by trying to change it.) If the change would alter the system state (the guess was incorrect), the operation is unauthorized and an error message might be produced, but if no actual state change would result, then the operation would not be affected by the access-secure assertion. Thus, the presence of the error message could yield some information. As published, the formal model prohibits any state change at all (including displaying an error message), so this channel is denied. A practical implementation can easily provide error

messages and still deny this channel by checking access sets before invoking any command – the user gets an error message if the appropriate entries are not in the access set regardless of whether the requested operation would alter the system state. Representing the sequential nature of this checking turns out to be difficult in the chosen formalism, but because implementing checks of this kind is so straightforward, the omission of error messages from the formalism is not a significant problem.

5. Adapt When Necessary

The intent of the informal model is to describe the system and the security model in terms accessible to users. "Operations" as defined in the informal model are intended to reflect the actual commands issued by users. Considering actual message systems and the development of the rapid prototypes, however, led us to conclude that user-visible commands frequently suppress some operands in order to provide an acceptable user interface. For example, an operation that creates a new message file may also change some other entities in the system (e.g., the directory that contains the file).

The restriction against side effects in the formal model discussed in the Section 3 would require that all entities referred to by a user-visible operation be explicitly included as operands, if "operation" in the formal model were directly mapped to "user-visible operation" From this observation, we concluded that user-level, complex operations must be mapped into the simpler vocabulary of security model operations. This was a significant departure from our original approach. The consequence has been two levels of specification: one for user-visible operations (ICL commands) and for each such command, a decomposition into formal security model operations. Development of these specifications is now proceeding in parallel with work on the rapid prototypes.

The lesson here is simply to beware of hidden assumptions in an informal model and be ready to adapt to them when they are revealed.

6. Summary

Lessons to be learned from our experience:

-stating even a carefully developed informal security model in mathematical notation can be a substantial undertaking;

-it is important to keep not only the informal model but also the application itself strongly in mind when building the formal model;

-some properties that seem easy to assure in an implementation can be hard to specify in a security model; and

-be ready to adapt when hidden assumptions become apparent.

7. Acknowledgment

The lessons documented here were learned only in the course of lengthy (and occasionally heated!) discussions of proposed formalizations with John McLean and Constance Heitmeyer. Without their efforts this note could not have been written.

8. References

Corn84 Cornwell, M., and Jacob, R. J. K., Structure of a rapid prototype Secure Military Message System. Proc. 7th DoD/NBS Computer Security Conference, Gaithersburg, MD, 24-26 Sept. 1984.

Heit82 Heitmeyer, C.L., Landwehr, C.E., and Cornwell, M., "The use of quick prototypes in the Secure Military Message Systems project," Proc. ACM SIGSOFT Second Software Engineering Symposium: Workshop on Rapid Prototyping, April, 1982, Columbia, MD. Reprinted in ACM SIGSOFT Software Engineering Notes, Vol. 7, No. 5 (Dec. 1982) pp. 85-87.

Land80 Landwehr, C.E. Assertions for verification of multi-level secure military message systems. Workshop on Formal Verification, SRI, Menlo Park, CA, April 1980. Reprinted in ACM SIGSOFT Software Engineering Notes, Vol. 5, No. 3 (July 1980) pp.46-47.

Land84 Landwehr, C.E., Heitmeyer, C.L., and McLean, J., A Security Model for Military Message Systems. ACM Trans. on Computer Systems, Vol. 2, No. 3, (August 1984) pp. 198-222.

Pope78 Popek, G.J., and D.A. Farber. A model for verification of data security in operating systems. CACM 21, 9 (Sept. 1978) pp. 737-749.