A FORMAL STATEMENT OF THE MMS SECURITY MODEL

John McLean
Carl E. Landwehr
Constance L. Heitmeyer

Computer Science and Systems Branch
Information Technology Division
Naval Research Laboratory
Washington, DC 20375

ABSTRACT

To provide a firm foundation for proofs about the security properties of a system specification or implementation, a formal statement of its security model is needed. This paper presents a formal model that corresponds to an informal, application-based security model for military message systems (MMS) that has been documented elsewhere. Following the formal statement, some considerations that led to its present form are discussed. The paper concludes with the statement of a "Basic Security Theorem" for the model.

1. Introduction

The Military Message System (MMS) security model [4] comprises 15 definitions, a one-paragraph description of MMS operation, 4 assumptions about user behavior, and 10 assertions that hold for the MMS. We focus on formalizing the 10 assertions (shown in Figure 1) only, although in doing so, some notation is required to define formal entities that correspond to those discussed informally in the 15 definitions. Below, the assertions are explicated formally in definition (2) concerning system states and definitions (5) through (11) concerning the system transform. Although the correctness of the explication cannot be proven, we discuss the correspondence between the formalism and the informal model briefly following the explication.

Each MMS family member can be modeled as an automaton with inputs, an internal state, and outputs. The inputs correspond to the commands users give to the system. Because this is a *security* model, we are principally concerned with modeling the categories of inputs that affect system security. The internal state of the automaton corresponds to the information currently stored in the message system -- messages, message files, classifications, access sets, and so on. Output from the automaton consists of command responses -- the things that users view or obtain in response to particular requests. These may include entities, parts of entities, classification labels, and IDs. We model output as a set of distinguished entities; although output is treated as part of the internal state, it represents that part that is directly visible to users. Some commands cause a state change that affects the output set, others may cause a change of state without changing the output set, and still others (particularly commands that do not satisfy the security assertions) cause no state change at all. A history of the system is a particular sequence of inputs and states.

2. Approach

We assume the existence of a set of possible users and a set of possible entities. Given these sets we define *system state* and the notion of a *secure state*. Next, we define *system* and *history* and introduce constraints on the transform that moves a system from one state to another. A system whose transform meets all these constraints is said to be *transform secure*. Finally, the notions of *secure history* and *secure system* are defined.

The structure of the formal model is intended to simplify its application to defining pre-conditions and post-conditions for system operations. To make explicit the entities that a given operation may change, we define the concept of *potential modification* based, in part, on the work of Popek and Farber [6]. Potential modification is similar to *strong dependency*, developed by Cohen [2].

3. System State

In this section we define what it is to be a system state and what it is for a system state to be secure. We assume the existence of the following disjoint sets.

OP is a set of operations.

L is a set of security levels. $\geq$ is a partial order on L such that $(L,\geq)$ is a lattice.

Authorization	1.	A user can only invoke an operation on an entity if the user's userID or current role appears in the entity's access set along with that operation and with an index value corresponding to the operand position in which the entity is referred to in the requested operation.
Classification hierarchy	2.	The classification of any container is always at least as high as the maximum of the classifications of the entities it contains.
Changes to objects	3.	Information removed from an object inherits the classification of that object. Information inserted into an object must not be have a classification higher than the classification of that object.
Viewing	4.	A user can only view (on some output medium) an entity with a classification less than or equal to the user's clearance and the classification of the output medium. (This assertion applies to entities referred to either directly or indirectly).
Access to CCR entities	5.	A user can have access to an indirectly referenced entity within a container marked "Container Clearance Required" only if the user's clearance is greater than or equal to the classification of that container.
Translating indirect references	6.	A user can obtain the ID for an entity that he has referred to indirectly only if he is authorized to view that entity via that reference.
Labeling requirement	7.	Any entity viewed by a user must be labeled with its classification.
Setting clearances, role sets, device levels	8.	Only a user with the role of System Security Officer can set the clearance and role set recorded for a userID or the classification assigned to a device. A user's current role set can be altered only by that user or by a user with the role of System Security Officer.
Downgrading	9.	No classification marking can be downgraded except by a user with the role of downgrader who has invoked a downgrade operation.
Releasing	10.	No draft message can be released except by a user with the role of releaser. The userID of the releaser must be recorded in the "releaser" field of the draft message.

Figure 1: Assertions of the MMS Security Model.

UI is a set of userid's.

RL is a set of user roles.

US is a set of users. For all $u \in US$, $CU(u) \in L$ is the clearance of u, $R(u) \subseteq RL$ is the set of authorized roles for u, and $RO(u) \subseteq RL$ is the current role set for user u.

RF is a set of references. This set is partitioned into a set, DR, of direct references and a set, IR, of indirect references. Although the exact nature of these references is unimportant, we assume that the direct references can be ordered by the integers. In this model we treat each direct reference as a unary sequence consisting of a single integer, e. g., <17>. Each indirect reference is treated as a finite sequence of two or more integers, e. g., $<n_1, \cdots, n_m>$, where $<n_1>$ is a direct reference.

VS is a set of character strings. These strings serve primarily as entity values (e. g., file or message contents).

TY is a set of message system data types that includes "DM" for draft messages and "RM" for released messages.

ES is a set of entities. For all $e \in ES$

$CE(e) \in L$ is the classification of e. $CCR(e)$ is ***true iff*** e is marked CCR, else ***false***. $AS(e) \subseteq (UI \cup RL) \times OP \times N$ is a set of triples that compose the access set of e. $(u, op, k) \in AS(e)$ ***iff*** u is a userid or user role authorized to perform operation op with a reference to e as op's kth parameter. $T(e) \in TY$ is the type of entity e. $V(e) \in VS$ is the value of entity e. If $T(e) = DM$ or $T(e) = RM$, then $V(e)$ includes a releaser field $RE(e)$, which if nonempty, contains a userid.

ES contains as a subset the set of entities that are containers. For any entity e in this set $H(e) = <e_1, \cdots e_n>$ where entity e_i is the ***ith*** entity contained in e. The set O of output devices is a subset of the set of containers.[1] Elements $o \in O$ serve as the domain of two further functions. $D(o)$ is a set of ordered pairs $\{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$ where each y_i is displayed on o. Each x_i is either a user or an entity, and the corresponding y_i is either a reference or the result of applying one of the above functions to x_i.[2] We require that $(x, V(x)) \in D(o) \rightarrow x \in H(o)$.[3] $CD(o)$ gives the maximum classification of information that may be displayed on o. This allows $CE(o)$ to be used as the current upper limit of the classification of information to be displayed by the output device, so that users can restrict the classification of output to be less than the maximum level permitted.

A state maps a subset of userids and references (intuitively, those that exist in the state in question) to elements of US and ES that represent their corresponding properties. A state also maps a subset of userids that "exist" into references that correspond to output devices (intuitively, these users are logged on to the specified devices). To this end we define three mappings. An ***id function***, U, is a one-to-one mapping from a (possibly improper) subset of UI into US. A ***reference function***, E, is a mapping from a (possibly improper) subset of RF into ES such that for all $n \geq 2$ $E(<i_1, \cdots, i_n>) = e$ ***iff*** $E(<i_1, \cdots, i_{n-1}>) = e^*$ where e^* is a container such that e is the i_n***th*** element of $H(e^*)$. For any reference r, if $E(r) = e$, we say that r is a reference to e (relative to E). A ***login function***, LO, is a one-to-one mapping from a (possibly improper) subset of UI into RF.[4]

Given a reference function, E, each indirect reference of the form $<n_0, \cdots, n_m>$ to an entity e_m corresponds to a path of entities $<e_0, \cdots, e_m>$ such that each $e_i \in rng(E)$, e_0 is denoted by the direct reference $<n_0>$, and for all positive integers $i \leq m$, e_i is the n_i***th*** entity in container e_{i-1}. Such an indirect reference is said to be ***based on*** each entity e_j where $0 \leq j < m$.

Definition (1): A ***system state***, s, is an ordered triple[5] (U, E, LO) where U is an id function, E is a reference function, and LO is a login function such that $dom(LO) \subseteq dom(U)$ and $rng(LO) \subseteq dom(E \cap (RF \times O))$. We also require that if $o \in rng(E) \cap O$ and $(x, y) \in D(o)$, then $x \in rng(E) \cup rng(U)$ to assure that only information about users and entities that "exist" in the current state can actually be displayed, and that for any reference r, $(x, r) \in D(o) \rightarrow E(r) = x$. Finally, we require that $E(LO(u_1)) = E(LO(u_2)) \rightarrow u_1 = u_2$ to prevent two users from being logged in simultaneously on the same terminal.

Given a system state $s = (U, E, LO)$, we abbreviate $E(r)$ by r_s, $U(u)$ by u_s, and $E(LO(u))$ by $\hat{u}_s$.

Definition (2): A state s is ***secure*** if $\forall\, x, y \in rng(E)$, $\forall o \in O \cap rng(E)$, $\forall w \in dom(LO)$, and $\forall u \in rng(U)$:

$$x \in H(y) \rightarrow CE(x) \leq CE(y),$$
$$x \in H(\hat{w}_s) \rightarrow CU(w_s) \geq CE(x),$$
$$(x, V(x)) \in D(o) \rightarrow (x, CE(x)) \in D(o),$$
$$RO(u) \subseteq R(u), \text{ and}$$
$$CD(o) \geq CE(o).$$

4. Secure System

In this section we define what a system is and what it is for a system to be secure.

Definition (3): A ***system*** Σ is a 4-tuple (I, S, s_0, T) where

1. In implementations, some kinds of output "disappear" from the system state (e.g., information sent to a printer or a telecommunications port) while others persist (e.g., information displayed on the screen of a terminal, which a user may later refer to and modify). In the formalization, we do not distinguish between these types; both are intended to be covered by O.

2. Both the item and what is displayed must be specified so that cases in which, for example, two entities have identical values but different security levels, can be distinguished.

3. We extend the set theoretic notions of membership and intersection to apply to tuples in the obvious sense.

4. The condition that LO is a function reflects an assumption that a user cannot be on two terminals at the same time. This assumption is merely for ease of exposition.

5. State is defined as a tuple, rather than as a set of functions, because two states whose elements have the same values are in fact identical, while two entities for which the defined functions return the same values may in fact be different (e.g., two copies of the same citation).

I is the set of well-formed system requests, where each request $i \in I$ is of the form $<op, x_1, x_2, \cdots, x_n>$ where each $x_j \in RF \cup UI \cup VS$ and $op \in OP$;

S is the set of possible system states;

s_0 designates a special state called the *initial state*; and

T is the *system transform*, i.e., a function from $UI \times I \times S$ into S.

Definition (4): A ***history***, Π, of a system is a function from the set of nonnegative integers N to $UI \times I \times S$ such that (1) the third element of $\Pi(0)$ is s_0, and (2) for all $n \in N$, if $\Pi(n)=(u,i,s)$ and $\Pi(n+1)=(u^*,i^*,s^*)$, then $T(u,i,s)=s^*$.

Before defining what it means for an operation to potentially modify an entity, we notice that a reference function E, and *a fortiori* a state S, induces a set of functions defined on references that are counterparts to the set of functions introduced above that are defined on entities. For example, there is a function, call it V_s, such that $V_s(r)=V(r_s)$. Similarly, there is a counterpart relation, call it H_s, such that $H_s<r, r^1, \cdots, r^n>$ ***iff*** $H(r_s)=<r_s^1, \ldots, r_s^n>$. Each counterpart is the user-visible version of the corresponding entity function. We call these counterparts, ***referential counterparts*** and use them to define what it means for two states to be equivalent except for a set of references.[6]

States $s=(U,E,LO)$ and $s^*=(U^*,E^*,LO^*)$ are ***equivalent except for*** some set of references S ***iff*** (1) $U=U^*$, (2) $LO=LO^*$, (3) $dom(E)=dom(E^*)$, (4) for any entity function F except V, $F_s=F_{s^*}$, and (5) for any reference $r \in dom(E) \sim S$, $V_s(r)=V_{s^*}(r)$.

We now define potential modification as follows:

u,i,s ***potentially modify*** r ***iff*** $\exists\, s_1, s_1^*$: s_1 is equivalent to s except possibly for some set of references and $T(u,i,s_1)=s_1^*$ and $V(r_{s_1}) \neq V(r_{s_1^*})$.[7]

Call y a ***contributing factor*** in such a case ***iff*** $\exists\, s_1$ as above and s_2, s_2^*: s_1 and s_2 are equivalent except for $\{y\}$ and $T(u,i,s_2)=s_2^*$ and $V(r_{s_2^*}) \neq V(r_{s_1^*})$.

That is, u,i,s potentially modifies r if there is some (second) state that may differ from s in the values of some entities, and T maps u,i, and this state into a third state in which r's value differs from that which it had in the second state. The contributing factors are those entities whose values affect r's final value.

6. We could have developed the entire formal model in terms of referential counterparts, but preferred the simplicity of functions to working with the relations H_s and LO_s.

7. This covers cases of creation (and deletion) since $V(r_{s_1})$ will be undefined and $V(r_{s_1^*})$ will be defined (although possibly empty).

For each referential counterpart and each function defined on users, we posit a unique operation that changes an entity or user with respect to that function. For example, an operation *set_AS*$(r,$*new_access_set*$)$ is the *only* operation that affects r's access set, and it has no other user-visible effect. Further, if the transition is, e. g., from state s to state s^*, $AS_{s^*}(r)$ is *new_access_set* if *new_access_set* is a character string and $V_s($*new_access_set*$)$ if *new_access_set* is an entity reference. Changes to the domain of E or U (creation or deletion of entities or users) are also assumed to occur only by explicit request. The formal release operation defined below is the single exception to this assumption; it changes the *type* of r and, potentially, the releaser field of r's value as well.

The exact nature of these operations is unimportant since these assumptions are included solely for ease of exposition. Their purpose is not to rule out implementation commands that affect different parts of entities, but to eliminate the problem of unspecified side effects in the formal model (e.g., permission to view a message marked *CCR* is not permission to clear the *CCR* mark). Implementation commands that can alter more than a single part of a single entity correspond to a sequence of formal operations. For a given implementation, this correspondence is determined by the semantics of the implementation command language. Once this correspondence has been determined, so that the security-relevant effects of each user command are clear, I can be replaced by the set of implementation commands with access sets also changed accordingly. Nevertheless, prudence dictates that modifications that can be made only by the *security officer* (e. g., changing a user's clearance), be restricted so that there is only a single command that performs them in any implementation.

The following constraints on the system transform lead to the definition of a *secure history* and a *secure system*. Where quantification is not explicit below, universal quantification is assumed.

Definition (5): A transform T is *access secure* ***iff*** $\forall u,i,s,s^*$: $T(u,i,s)=s^*$, $[(op \in i \cap OP$ and $r_k \in i \cap RF) \rightarrow ((u,op,k) \in AS(E(r_k))$ or $\exists l \in RO(u_s)$ and $(l,op,k) \in AS(E(r_k)))]$ or $s=s^*$.[8]

8. For simplicity we disregard *error messages* in the formalism. In an implementation we assume that if an unauthorized operation is attempted, an appropriate error message will be produced in the next state.

Definition (6): A transform T is *copy secure iff* $\forall u,i,s,s^*$: $T(u,i,s)=s^*$, x is potentially modified with y as a contributing factor $\rightarrow CE(x_s) \geq CE(y_s)$.

Definition (7): A transform T is *CCR secure iff* $\forall u,i,s,s^*$: $T(u,i,s)=s^*$, $r \in i \cap IR$ is based on y and $CCR(y)$ and z is potentially modified with r as a contributing factor $\rightarrow CU(u_s) \geq CE(y)$.

Definition (8): A transform T is *translation secure iff* $\forall u,i,s,s^*$: $T(u,i,s)=s^*$, $x \in DR$ and $(x_{s^*},x) \in D(\hat{u}_{s^*}) \rightarrow \exists\, r \in i \cap RF$, $r_s = x_s$ and (r is based on z and $CCR(z) \rightarrow CU(u_s) \geq CE(z)$).

Definition (9): A transform T is *set secure iff* $\forall u,i,s,s^*$: $T(u,i,s)=s^*$,
(a) $\exists o \in dom(E \cap (RF \times O))$, $CD(o_s) \neq CD(o_{s^*})$ or $\exists x \in dom(U)$, $CU(x_s) \neq CU(x_{s^*})$ or $R(x_s) \neq R(x_{s^*}) \rightarrow security\_officer \in RO(u_s)$; and
(b) $x \in dom(U)$ and $RO(x_s) \neq RO(x_{s^*}) \rightarrow u_s = x_s$ or $security\_officer \in RO(u_s)$.

Definition (10): A transform T is *downgrade secure iff* $\forall u,i,s,s^*$: $T(u,i,s)=s^*$, $x \in dom(E \sim (RF \times \{\hat{u}_s\}))$ and $CE(x_s) > CE(x_{s^*}) \rightarrow downgrader \in RO(u_s)$.

Definition (11): A transform T is *release secure iff* $\forall u,i,s,s^*$: $T(u,i,s)=s^*$, $(T(x_s)=RM \rightarrow T(x_{s^*})=RM$ and $RE(x_{s^*})=RE(x_s))$
and $(T(x_s) \neq RM$ and $T(x_{s^*})=RM \rightarrow RE(x_{s^*})=u$, $\exists r$: $r_s = x_s$, i is the operation $<release, r>$, $releaser \in RO(u_s)$ and $T(x_s)=DM)$.

Definition (12): A transform is *transform secure iff* it is access secure, copy secure, CCR secure, translation secure, set secure, downgrade secure, and release secure.

Definition (13): A history is *secure* if all its states are state secure and its transform is transform secure.

Definition (14): A system is *secure* if each of its histories is secure.

5. Discussion

Perhaps the most basic decision we made in formalizing the MMS model concerned our general conception of a computer system, in particular the relation between a system state and a system. We considered a view where a system state consists of entities and their relations, and a system adds to this users and user operations on entities. Hence, all restrictions on user properties (in particular, the restriction that for all u, $RO(u) \subset R(u)$) are included in the definition of a secure system. We chose instead to view the distinction between system states and systems in terms of static as opposed to dynamic properties. Static properties are those that hold of all secure states and hence, can be checked by examining a state in isolation; dynamic properties are those that need only hold for the relation between secure states and hence, can be checked only by comparing two or more states. In the view we adopted, all static security properties are included in the definition of a secure state.

To a large extent the choice in conceptualizations is a matter of taste. Bell and LaPadula [1] use the latter, while [3] leans to the former. By minimizing the notion of a secure state, the former view makes the Basic Security Theorem shorter. The deciding factor in our adopting the latter view is that it makes it impossible for a system to undergo a security-relevant change without undergoing a change in state.

Principal difficulties we encountered in formalizing the MMS security model were in representing "copy" and "view", system output, and the notion of an authorized operation. Assertion 3 ("changes to objects") in the informal model requires formal semantics to reflect the movement of information between entities, while assertion 4 ("viewing") requires formal semantics to reflect making an entity visible to a user. Assertion 5 ("accessing CCR entities") now addresses both copying and viewing. The semantics for "copy", embodied in the definitions of "potential modification" and "contributing factor", are based on a broad interpretation of "copy." Information is considered to be copied, not only if it is directly moved from one entity to another, but also if it contributes to the potential modification of some other entity. For example, if an operation scans message file A and copies messages selected by a filter F to message file B, both A and F contribute to the potential modification of B (and are therefore subject to the constraints imposed by *copy secure* and *CCR secure*), even if both A and F are empty. The semantics for "view" are straightforward: a thing is viewed if an operation makes it a member of an output container. In light of these considerations, we have broadened assertion 5 from earlier drafts to use "access" in place of "view".

In the formalization, system output is interpreted as a set of containers; other entities, parts of entities, references, and classifications that are made visible to a user are interpreted as being copied to his output container. We assume that in any implementation the classifications displayed appear close to the entities (or parts) they correspond to, but we have not formalized this assumption. References are explicitly included as a part of output because the same operation applied to the same entities can yield different results depending on how the entities are referenced. This leads to the constraint (*translation secure*) on operations that produce as output direct references that are translations of indirect ones. To enforce this constraint, the system must recognize references as a particular kind of output.

Formalizing the concept of an authorized operation is difficult because the semantics of authorized operations are unspecified. Our definition of *access secure* requires that, if an operation changes the system state or produces output (beyond an error message), then for each entity in the set of operands, the user or role, operation, and operand index must appear in the access set. Unauthorized operations must not alter the system state except to report that they are erroneous.

6. Correspondence to the Informal Model

Assertions (2), (4), and (7) of the informal model, concerning classification hierarchy, viewing, and labeling, are incorporated in the formal definition of *secure state*. They correspond respectively to the first three conditions a secure state must satisfy; the last two conditions require that each user's current role set must be a subset of his authorized role set and that the current security level of each output device must be dominated by its maximum allowed level. These last two conditions are implicit in the informal model.

The remaining assertions of the informal model have been translated into constraints on the system transform. Assertion (1) (authorization) corresponds directly to *access secure*, assertion (6) to *translation secure*, and assertions (8)-(10) (setting, downgrading, and releasing) correspond respectively to *set secure*, *downgrade secure*, and *release secure*. *Set secure* has been expanded from earlier drafts to restrict the setting of device classifications and user role sets to security officers and to restrict the setting of a user's current role to himself or a security officer. *Downgrade secure* contains an exception for $\hat{u}_s$ so that a user is not prohibited from lowering the current level of his output device. The formal statement of *release secure* makes explicit the requirement that, once released, a message cannot have its type or releaser field altered.

Assertions (3) and (5) correspond to *copy secure* and *CCR secure*. The definition of *copy secure* actually covers parts of both assertion 3 and assertion 4 because output devices are treated as a set of containers. So, if entity x receives information from an object y, $CE(x) \geq CE(y)$ (assertion 3), and that if an output container o receives information from entity x, $CE(o) \geq CE(x)$ (assertion 4). *CCR secure* corresponds to assertion 5, under the interpretation that having access to an entity is significant only if that entity is a contributing factor in the potential modification of another entity.

7. Storage Channels

Because we have defined "potential modification" and "contributing factor" in terms of changes only to the *value* of an entity, the constraints imposed by *copy secure* and *CCR secure* do not apply to changes made to other functions defined on entities (classification, CCR mark, access set, and type) or those defined on users (clearance, role set, and current role). Thus, there is the potential for information to be transferred from a higher level to a lower one through these functions. However, changes to user clearances and role sets are controlled by *set secure*; changes to classifications are controlled by *downgrade secure*; changes to entity type will generally be restricted severely by the semantics of the message system; and changing a CCR mark provides only a single bit of information at a time. Changes to the access set for an entity could provide a higher rate of information transfer, and if this were a serious concern, the definitions in question could be modified to include changes of this kind. We chose not to include these because our principal concern is with changes to the values of entities, and the added notational complexity would only cloud the presentation. We have left for others the problem of dealing with classifications that are themselves classified, such as highly sensitive compartment names.

8. A Basic Security Theorem for the Formal MMS Security Model

In formalizations where a secure system is a collection of secure states, some feel that a Basic Security Theorem is needed to show the restrictions on system transforms that insure that a system starting in a secure state will not reach a state that is not secure.[9] Such theorems are of little practical significance, since their proofs do not depend on the particular definition of security provided by the model [5]. Further, in our approach such a theorem is not pressing since the concept of a secure system is defined largely in terms of a secure transform. Nevertheless, we do appeal to the notion of a secure state, and some readers may feel that some form of Basic Secu-

9. See for example [1].

rity Theorem is needed. Those readers should find it trivial to prove the following analog of the Basic Security Theorem for our definition of a secure state.

Theorem: Every state of a system Σ is secure if s_0 is secure and T meets the following conditions for all u, i, s, s^*: $T(u,i,s)=s^*$ and for all $x, y \in RF$, $w \in US$:

1. $x_s \notin H(y_s)$ and $x_{s^*} \in H(y_{s^*}) \rightarrow CE(x_{s^*}) \leq CE(y_{s^*})$.
2. $x_s \in H(y_s)$ and $CE(x_{s^*}) \not\leq CE(y_{s^*}) \rightarrow x_{s^*} \notin H(y_{s^*})$.
3. $x_s \notin H(\hat{w}_s)$ and $x_{s^*} \in H(\hat{w}_{s^*}) \rightarrow CU(w_{s^*}) \geq CE(x_{s^*})$.
4. $x_s \in H(\hat{w}_s)$ and $CU(w_{s^*}) \not\geq CE(x_{s^*}) \rightarrow x \notin H(\hat{w}_{s^*})$.
5. $(x_s, V(x_s)) \notin \hat{w}_s$ and $(x_{s^*}, V(x_{s^*})) \in \hat{w}_{s^*} \rightarrow (x_{s^*}, CE(x_{s^*})) \in \hat{w}_{s^*}$.
6. $(x_s, V(x_s)) \in \hat{w}_s$ and $(x_{s^*}, CE(x_{s^*})) \notin \hat{w}_{s^*} \rightarrow (x_{s^*}, V(x_{s^*})) \notin \hat{w}_{s^*}$.
7. $R(w_s) \neq R(w_{s^*})$ or $RO(w_s) \neq RO(w_{s^*}) \rightarrow RO(w_{s^*}) \subseteq R(w_{s^*})$.
8. $CE(\hat{w}_s) \neq CE(\hat{w}_{s^*})$ or $CD(\hat{w}_s) \neq CD(\hat{w}_{s^*}) \rightarrow CD(\hat{w}_{s^*}) \geq CE(\hat{w}_{s^*})$.

Together, (1)-(8) are necessary and sufficient conditions for every state of a system to be secure in any system that does not contain states that are unreachable from s_0.

Acknowledgements

Jon Millen's helpful comments on two drafts of this paper led to substantial improvements in it.

References

[1] D. E. Bell and L. J. LaPadula, "Secure Computer System: Unified Exposition and Multics Interpretation," M74-244, MITRE Corp., Bedford, MA, July, 1975.

[2] Ellis Cohen, "Information Transmission in Computational Systems," *Proc. 6th ACM Symp. Operating Systems Principles, ACM SIGOPS Operating System Rev.*, Vol 11, No. 5 (Nov. 1977) pp. 133-139.

[3] R. J. Feiertag, K. N. Levitt, and L. Robinson, "Proving Multilevel Security of a System Design," in *Proc. 6th ACM Symp. Operating Systems Principles, ACM SIGOPS Operating System Rev.*, Vol. 11, No. 5 (Nov. 1977) pp. 57-65.

[4] C. E. Landwehr, C. L. Heitmeyer, and J. McLean, "A Security Model for Military Message Systems," NRL Report, to appear fall, 1983. (Also submitted for external publication.)

[5] J. McLean, "A Comment on the Basic Security Theorem of Bell and LaPadula," submitted for publication, 1983.

[6] G. J. Popek and D. A. Farber, "A Model for Verification of Data Security in Operating Systems," *Communications of the ACM.*, Vol. 21, No. 9 (Sept. 1978) pp. 737-749.