

AN ALTERNATIVE IMPLEMENTATION OF THE REFERENCE MONITOR CONCEPT

G. KING W. SMITH

Magnavox Electronic Systems Company, NVSG
120 Ashburn Road Ashburn, Va. 22011

ABSTRACT

During the last decade, several research projects for multilevel secure automated military message systems have implemented the reference monitor concept with Bell and LaPadula security models. The result has been implementations, security kernels, that are characterized by poor performance and a resulting security model which does not accurately describe the real behaviour of a military message system. These facts have led Magnavox to pursue an alternative implementation of the reference monitor concept. Our resulting product is the Military Message Embedded Executive [(ME2)], and its supporting hardware base, the Trusted Military Message Processor (TRUMMP).

INTRODUCTION

The Military Message Experiment (MME) was a joint research effort sponsored by the Navy, Defense Advanced Research Projects Agency (ARPA) and the Commander-in-Chief Pacific (CINCPAC) to produce and evaluate the feasibility of developing multilevel secure military message systems. That project produced SIGMA, an operational system that was used by military officers and staff personnel. Later evaluations of the system by the Naval Research Laboratory (NRL) demonstrated the serious deficiencies that arise when a military message system is implemented with a Bell and LaPadula model. For background, a brief summary of the NRL findings and their solutions are presented here. Our description of the NRL findings are taken from [1].

Reference [1] argues that a security policy should enable users to understand how to operate the system effectively, implementors to understand what security controls to build, and certifiers to determine whether controls are consistent with directives and whether controls are correctly implemented. Reference [1] proceeds to describe three deficiencies of the Bell and LaPadula model that prevented the model from providing this guidance to SIGMA users,

implementors, and certifiers. Those deficiencies, as described in [1], are as follows:

Prohibition of write-downs. The *-property prohibits the writing down of information to a lower classification level. However, under some circumstances this action would be secure for a military message system. It was assumed that user confirmation by SIGMA would prevent security violations when this action was needed but because so few understood the security policy (a phenomenon derived from the numerous exceptions forced by the nature of the application) users tended to always permit these actions without understanding why they had been questioned.

Absence of multilevel objects. The model recognizes only single-level objects when in reality, objects for a military message system are inherently multilevel. An example is the multiple paragraphs of a message, each of which may have a different classification. By treating a multilevel object as a single level object, some information is treated as more classified than it really is.

No structure for application dependent security rules. The model contains no structure for application dependent rules. A military message system typically must enforce some security rules that are unique to its application. An example is a rule that allows only users with release authority to invoke the release operation.

Reference [1] continues by stressing the deficiencies of approaches that attempt to fit an application on top of the general purpose Bell and LaPadula model. The need for application based security models, as opposed to Bell and LaPadula derivatives, is emphasized. Such a model is defined for a military message system. Our product may prove useful in the implementation of these types of models.

Currently, we are coordinating with the National Computer Security Center (NCSC) with respect to selection of the evaluation criteria to be used in a formal evaluation. Initially, our development was guided by the Department of

Defense (DoD) Trusted Computer System Evaluation Criteria (TCSEC) [7] but we quickly noted that our expected application, an embedded component of a military communications network, was very different than the stand-alone general purpose ADP system described in [7]. As the Trusted Network Interpretation (TNI) [8] became available, it became increasingly evident to the NCSC and Magnavox that evaluation as a TNI network component would be more appropriate.

Nevertheless, while we were pleased that a TNI evaluation would be useful in evaluating the network aspects of our product, we also desired a set of criteria that would address the unique security requirements of a tactical and embedded systems environment. As a result, while we look to the TNI for guidance, we are investigating what we believe to be a reasonable set of evaluation criteria for an embedded system.

At this time, we are in the process of deciding whether to forgo a TNI evaluation and work with the NCSC in the development of a set of evaluation criteria for embedded and tactical systems or to pursue a formal TNI evaluation. Either strategy has advantages and disadvantages.

This paper is organized into five major sections: (1) the functional and security requirements of military message systems; (2) an overview of attempts to implement these requirements with security kernels; (3) definition of a security kernel alternative; (4) implementation of this alternative in the Military Message Embedded Executive [(ME2)], including discussion of data and process security enforcement; and (5) our conclusions and future plans.

1. FUNCTIONAL AND SECURITY REQUIREMENTS

Systems that are used to process military messages are concerned with the handling of different message types. One type, the formal military message, is at the heart of Command, Control, Communications and Intelligence (C3I) systems. Nearly all military operations and policies are communicated through a formal military message [4]. Military standards which govern the format of these messages typically require a TO, FROM, INFO, DATE-TIME-GROUP, TEXT, SECURITY, and PRECEDENCE field for each message. A formal military message must be maintained for long periods of time and be capable of being quickly retrieved.

Another type of military message is the informal message. In contrast with formal messages, informal messages generally do not have the same storage and retrieval requirements.

The systems used to process these messages consist of both embedded processors and non-embedded processors. An embedded processor refers to those processors that do not contain a direct human-machine interface (HMI) but are

still a vital processing component (e.g. a message switch). While most non-embedded processors are concerned with the unauthorized disclosure or unauthorized modification of information to users (data security), the embedded processor must also enforce what is referred to as process security [3].

For an embedded military message processor, a process security requirement might be related to the precedence of the message. In a military message, the precedence specifies the maximum delivery time for a particular type of message. Consequently, a process security requirement for a military message system might state that higher precedence message processing will preempt lower precedence message processing. Enforcement of this process security requirement by the embedded computer allows the system to respond to high priority traffic as quickly as possible. In a C3I environment enforcement of this process security requirement is as important as the security requirements related to the unauthorized disclosure and unauthorized modification of data.

Finally, the objects of a military message system must reflect the multilevel nature of a military message. A military message is often composed of paragraphs of differing classification with the overall classification for the message equal to the highest classification of any part of the message. To treat the entire message as classified as the most sensitive portion causes some information to be treated as more classified than it really is. As a result, an information structure that is capable of representing the multilevel nature of a military message is required. This type of information structure is known as a "container" [1].

2. EXPERIENCE WITH SECURITY KERNELS

Until very recently the security kernel, a reference monitor implementation, has been promoted as an appropriate foundation for meeting nearly all security requirements. By enforcing a multilevel security policy, the security kernel creates an abstract machine upon which it is not possible for an application program to commit compromise. By restricting the role of security enforcement to a small and simple mechanism such as a security kernel, security verification is much more tenable.

In a military message environment, however, the first attempts to implement military message systems were faced with a practical problem related to the nature of the application. Although many of the required actions are commonly defined as secure, they violate the general purpose axioms of the security kernel (i.e. the *-property). An example is the classification of the message header information at security levels lower than that of the associated message text. To solve the problem, these systems relied on numerous trusted subjects which are effectively exceptions to the general purpose axioms.

Security kernel performance was an additional problem. Some kernels yielded performance as low as 10-25 percent of their non-trusted counterparts [2]. With good performance a critical requirement, security kernels did not yield adequate results. Yet another drawback of security kernels was their absence of attention to the process security requirements that are present in DoD embedded computers.

Thus, some alternatives for security enforcement in a military message system are: 1) develop a special purpose security kernel that will enforce a model tailored to military message systems, 2) use a Bell and LaPadula model with numerous trusted subjects, or 3) use a security kernel alternative [5]. The first alternative is extremely costly, and the second alternative yields a product with the problems that SIGMA experienced. As a result, TRUMMP and (ME2) use a security kernel alternative to implement application based security models specifically tailored to military message systems. Fundamental to our approach is the concept of a state machine architecture.

3. A SECURITY KERNEL ALTERNATIVE

The security kernel alternative that (ME2) will use to enforce an application security model is based on the concept of a network of communicating finite state machines (CFSM) that operate under the control of a State Machine Executive (SME) [5]. In our development the (ME2) is the state machine executive that provides domain concurrency, domain separation, inter-domain communication via message passing, as well as enforcement of application based data security and process security requirements.

3.1 Definitions

The following definitions are provided to establish the required terminology for discussion of the Military Message Embedded Executive (ME2).

Processing Nodes - The nodes of the graph in Figure 1. These nodes represent a processing domain. In the context of a military message system a processing node might represent the processing domain responsible for displaying a message while another node might represent the processing associated with the network interface.

Connections - The arcs of the graph in Figure 1. These arcs represent the communication channels between processing nodes. Connections may be external or internal. An internal connection refers to a connection that has a processing node for both its source and sink points. Connections that do not have a processing node for sink and source are external connections.

Source Station - The tail of each arrow in Figure 1. Data is sent from a source station to a sink station. A connection attached to a source station of a node is called a sink

connection for that node.

Sink Station - The head of each arrow in Figure 1. Data is received on a sink station from a source station. A connection attached to a sink station of a node is called a source connection for that node. Each sink station represents an unbounded FIFO queue, the tail of which acts as the sink point for source connections, while the head acts as a source station for sink connections.

Workstation - The heads of the queues associated with the sink stations of each processing node.

Container - A typed information structure that is categorized as single-level or multilevel based on its typed value. A container is composed of indivisible atomic elements known as objects.

Data Security Labels - Container labels which indicate the level of damage that might result if the container information is subjected to unauthorized disclosure or unauthorized modification.

Process Security Labels - Process labels and container labels that are used to enforce the process security requirements of an embedded computer.

4. THE MILITARY MESSAGE EMBEDDED EXECUTIVE [(ME2)]

The Military Message Embedded Executive [(ME2)], pronounced as "ME TWO", is an implementation of the state machine executive described in the previous section. The parenthesis around the acronym (ME2) are intentional to denote its embedded architecture. This section describes the (ME2) implementation of each of the state machine architecture concepts described above.

4.1 Processing Nodes

Processing nodes, also called logical processors, are the components of the state machine architecture that refer to processing segments which operate independently of each

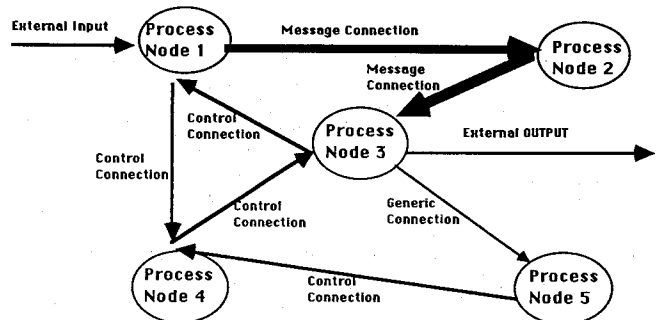


Figure 1. A State Machine Architecture

other. In a military message system, one processing node might represent the processing associated with display of a message while another node might represent the processing associated with a network interface. The fundamental requirement that must be supported in an implementation of a processing node is domain isolation. The resources of each processing node must be isolated. In a single computer implementation, this means that the (ME2) must insure that the computer registers, flags, memory, and code of one node are physically inaccessible to another node. Not only must the (ME2) establish domain isolation but it must also ensure that once established that the processing node cannot alter the domain isolation characteristics. In (ME2), the required domain isolation is accomplished, in large part, by the domain isolation support that the Trusted Military Message Processor (TRUMMP) provides.

The heart of the TRUMMP, an Intel 80286 microprocessor, is the foundation that the (ME2) depends on for establishing domain isolation at each processing node. The first feature of the Intel 80286 that the (ME2) uses to provide domain isolation is the Intel 80286 Task State Segment (TSS). The TSS defines the contents of all machine registers, flags, code, and memory that are physically visible at any one time. In the (ME2), each processing node has an associated TSS that completely defines the node's domain. Contained in the TSS is the address of another Intel 80286 data structure called the local descriptor table (LDT). The LDT is the TSS component that defines the memory resources and code segments that are associated with a particular processing node. Consequently, (ME2) controls the TSS and LDT of each processing node to provide domain isolation.

The Intel 80286 microprocessor was purposely chosen as the heart of the TRUMMP for its specific potential to provide an efficient implementation of domain isolation. Using the implementation described above, the (ME2) takes maximum advantage of firmware domain switching allowing a current domain to be saved, and a new domain established, in as little as 16 microseconds.

The second feature of the Intel 80286 that the (ME2) utilizes is the hierarchal protection rings. The hierarchal protection rings are used to insure that processing nodes cannot alter the domain that the (ME2) establishes. This protection is provided in two ways. First, because data at more privileged rings is hidden from data at less privileged rings, the TSS and LDT data structures are contained in the highest privilege ring making them invisible to all domains except the (ME2). Second, because the instructions that have the capability to change a domain may be executed only in the most privileged ring, only the (ME2) resides in the most privileged ring.

4.2 Connections

Connections, also called virtual channels, are the communication mechanism for process nodes. Controlled data flow over these connections is fundamental to the (ME2) enforcement of application based security models. An implementation of connections requires the (ME2) to enforce strong typing rules over the connection, as well as data and process security rules.

In the (ME2) implementation of connections, each connection has several associated attributes which describe the size of data that the connection accepts, the type (e.g. control, data, or message type) that the channel accepts, and whether or not the connection is internal or external. In addition the container has data and process security labels that must dominate the data and process labels of the sink before a transmission can occur. In the (ME2) implementation, the connections and their associated attributes are contained within a (ME2) data base that has been burned into Read Only Memory.

4.3 Source Stations, Sink Stations, and Workstations

At the (ME2) implementation level, source and sink stations are the heads and tails of data queues. Because the (ME2) provides first-in-first-out delivery of queued data, a source station is always the queue head while sink stations are always the queue tail. Workstation is a more general term that refers to a data depository at which a queue element (the sink station, the source station, or some other element) is accessible.

In the (ME2), a workstation is implemented as a permanent entry within a process node's local descriptor table. Effectively, this permanently defines the virtual address of a workstation. To access data contained at the data depository (i.e. a particular queue element), application programs access the virtual address of the workstation. As an analogy, consider the workings of a photographic slide projector. A queue of slides, present in the slide projector, are viewed individually by the projector's movement of a slide through the lens path. In much the same way, the (ME2) must make individual containers visible at a fixed virtual address for viewing by process nodes.

The descriptor table based memory management of the Intel 80286 microprocessor provides the efficient mechanism for making a new queue element visible at the workstation. Because the local descriptor table is actually a table of physical addresses and access rights associated with each virtual address, the (ME2) can efficiently make a new queue element visible at a workstation by copying the physical address of the queue element to the local descriptor table. As a consequence, the elements of a queue might be at many non-contiguous physical memory segments, but are viewed, because of the

actions of the (ME2), through a workstation at a fixed virtual address.

The (ME2) implements application based restrictions on workstation access according to the access rights declared for the queue. A workstation may be declared as read only, read-write, or no access. The (ME2) provides these access restrictions by writing the applicable value to the local descriptor table entry associated with the particular workstation.

In the (ME2) implementation, because each work station of a process node implies a unique local descriptor table entry for that process node, the number of work stations is physically limited by the maximum number of local descriptor table entries (approximately 8000). However, we expect the number of workstations to vary greatly based on the input and output requirements of the node. For example, a node responsible for message storage might segregate messages based on data and process security labels and thus require a large number of workstations, while another node might only require one workstation for an internal data structure.

4.4 Containers

As previously described, a container is broadly categorized as either a single-level or a multilevel information structure that is transmitted over a connection. Like the connections that they are transmitted over, containers have attributes that describe their size, type, data security, and process security attributes. Examples of container types include message containers, control containers, and data containers. Examples and a description of data security and process security attributes are provided in following sections.

An interesting feature of the (ME2) is the functional support provided for the transmission of multilevel containers. An example best illustrates this support. As an example consider two process nodes with differing secrecy authorizations.

The first process node, node A, is authorized to send and receive Top Secret (TS), Secret (S), and Confidential (C) information. As a result, it currently owns a multilevel container composed of individual objects classified at each of the authorized level (TS,S,C). There also exists a second process node, node B, with a need and authorization for some but not all of the information represented in node A's multilevel container. However, node B's secrecy authorizations are limited to secret and confidential data. In this instance, to allow transmission of the entire container, would be a security violation.

Nevertheless, as identified by the NRL evaluations of SIGMA, it would be useful if the less sensitive segments of the container could be transmitted. To allow this, the (ME2)

provides the PAGE, EXTRACT, and INSERT executive services.

The first of these services, the PAGE executive service, is a service that allows process nodes to define the "current" object within a container of objects. Consider a familiar analogy from the world of word processing. In much the same way as functions exist to position a cursor in a document, the PAGE function exists to position a marker on an atomic element, an object of a container. One less obvious side effect of the PAGE function is the fact that as the marker is moved to a new object of the container, the old object becomes invisible.

The additional services, EXTRACT and INSERT, provide a method for the owning process node to build a new, potentially multilevel, container for transmission to node B. These services perform the functions implied by their names. They accept containers identifiers as operands and operate on the "current" container object as identified by the previous PAGE requests. In the example, when these services are used by process node A to extract the less sensitive segments of its multilevel container, the problems associated with treatment of the entire container at the level of the most sensitive data are avoided.

We envision that specific applications may use these (ME2) services to build additional services that meet specific application requirements. An example would be a service that accepts a multilevel container as input and returns a single level subset of that container based on secrecy, integrity, and/or necessity attributes.

In the (ME2), containers are implemented according to user configurable allocations of physical memory based on container type and size. Consequently, a container is actually a contiguous segment of physical memory that contains data of the type declared for the container as well as data and process security labels. Potentially multilevel containers, such as message containers, contain a data and process security label for each individual unit of information.

The (ME2) distributes containers to workstations based on the value of a workstation attribute known as the auto-refill attribute. If a workstation is defined as an auto-refill workstation, the (ME2) will associate an empty container with the workstation at system start and at any future time that all workstation containers have been transferred over a connection. An empty container is defined as a container whose contents have been set to zero or some other innocuous value by the (ME2). At system start, all containers are empty. Later, used containers are emptied by the (ME2) and become available for reuse as they are sent from the application back to the (ME2).

4.5 Data Sensitivity Labels

In (ME2), data sensitivity labels are implemented with a secrecy component and an integrity component. The secrecy component of the label provides up to sixteen hierarchical levels and sixty-four non-hierarchical compartments within each level. The integrity component provides a means of protecting against unauthorized modification of data. Two levels, high and low, are supported. The assignment of values to a data sensitivity label differs depending on whether the data sensitivity label is for a container or a workstation.

The data sensitivity labels for workstations are static. They reside in the read-only portion of the (ME2) data base and result from a system design based on a state machine architecture. Because the system design is based on isolating data flows and processing, the set of possible data security labels that a workstation may accept is known pre-runtime.

On the other hand, because containers are reusable, the values for the data sensitivity labels of a container must be assigned as the container is used. Initially, at the time the (ME2) provides a container to an auto-refill workstation, the data sensitivity labels of the container have a value known as obscure. That is, the (ME2) and all process nodes recognize that the container in question has not been labelled. Certain process nodes may request the (ME2) to change the value of a data sensitivity label for a container from obscure to some other value. The (ME2) will perform such a label change if and only if such a change is consistent with the process security requirements. That is, just as only certain individuals have the authority to downgrade a military message, the (ME2) ensures that only certain process nodes may change a container label.

From a (ME2) implementation perspective, data sensitivity labels are always contained in memory that is accessible only to the (ME2). Consequently, data sensitivity labels are only changeable if the process node is trusted and the change is requested through the appropriate (ME2) service request.

4.6 Process Sensitivity Labels

Process security requirements ensure the prevention of undesirable and potentially catastrophic events in an embedded computer system. A general example is the requirement that a certain weapon system be fired only after a sequence of controlled events has been implemented. An example taken from military message system domains is the requirement that the processing of more important messages preempt the processing of less important messages or the requirement that among equal priority messages, messages should be processed in a first-in-first-out (FIFO) manner.

The Naval Research Lab suggests that one approach for enforcing process security

requirements is the approach that is currently used for enforcing data security requirements. Specifically, the NRL suggests that to enforce process security, a system be structured into a set of functions that affect process security and a set that do not [3]. By verifying that the functions that have the capability to violate process security do not violate it (they are trusted), process security can be assured.

This approach is the approach we have taken with the (ME2). As a result, the (ME2) is the function that enforces data security and process security. As mentioned, an important process security requirement of most military message systems is the requirement that the processing of more important messages preempt the processing of less important messages and that among equal priority messages, messages are processed in a first-in-first-out (FIFO) manner. Just as the (ME2) provides sensitivity labels to enforce data security requirements, a similar label is needed to designate the relative importance of a message so that process security requirements can be enforced. In the (ME2), this label is referred to as the container's "Necessity". Five necessity values are currently supported in the (ME2).

By assigning necessity labels to each container, containers may be segregated allowing higher priority message processing containers to be maintained on queues which do not also contain lower priority messages. This eliminates the need for a shared queue of multiple necessity levels. As with data security labels, the (ME2) uses the necessity labels to restrict the flow of containers over connections. The (ME2) will transfer a container over a connection only if the necessity label of the container dominates the necessity label of the sink. Consequently, the segregation of containers coupled with the (ME2) process node preemption based on the arrival of a container at an empty workstation provides assurance that the data that has arrived is of greater importance than current processing, and that the arrival will cause preemption. This enforces the process security requirement that higher priority message processing preempt lower priority message processing.

The other type of process security label that the (ME2) provides is a command label which lists all of the authorized (ME2) commands that a process node can request. An earlier example which describes how the command label is used to prohibit all nodes except authorized nodes from changing the data security labels of a container is the best illustration.

4.7 Audit Requirements

In the (ME2), because the intended operational environment of the TRUMMP is as an embedded component of a large telecommunications network (see Figure 2), provisions for auditing capabilities require special attention. As stated in the TCSEC and TNI, audit requirements

include the ability to audit specific events such as the use of identification and authentication mechanisms by specific terminals, actions taken by computer operators, and the override of human-readable output markings for printed material.

However, because the (ME2) does not directly communicate with printers, terminals, secondary storage devices, or computer operators, the specific (ME2) audit requirements require a unique interpretation that will support the TCSEC and TNI audit objectives when the system is considered as a whole. The (ME2) approach to auditing is provided by an application defined audit manager that makes use of application information and low-level audit information available from the (ME2).

The low-level audit information available from the (ME2) includes information concerning the use of all connections, the dispatching of process nodes, the receipt of interrupts, attempts to modify the connection table, and the time at which these events occurred. Information is provided to the application audit manager through a well defined, software callable, audit interface.

It is anticipated that the information provided by this (ME2) interface, the (ME2) real-time clock, and application information will be sufficient for the construction of audit records to be exported to an audit component of the network.

4.8 The (ME2): A Summary

The Military Message Embedded Executive (ME2) is an executive which also contains a security enforcing foundation based on the data and process security requirements of military message systems. The key to security in the (ME2) is a state machine architecture which divides the system into process nodes. Information is stored in strongly typed containers which are transferred over connections to strongly typed workstations. The (ME2) implements traditional data security labels as well as additional labels useful in the enforcement of process security requirements. The deficiencies of SIGMA, a security kernel for the Military Message Experiment (MME), do not exist. Specifically, (ME2) provides the capability for authorized downgrade, implements multilevel objects (containers), and provides a structure for implementation of application dependent security requirements.

5. CONCLUSIONS, STATUS, AND FUTURE PLANS

This paper has described the progress of Magnavox research into the multilevel secure automated exchange of military messages. This work and that of the Naval Research Lab represents new approaches to "designed in

security" that are not based on the security kernel and Bell/LaPadula model approaches that have dominated military message systems and the industry for the past fifteen years. Instead, the approach is based on the concept of a network of communicating finite state machines. The (ME2) is an efficient implementation of this concept with specific features that appear useful in the design and implementation of military message systems.

Beyond its state machine architecture, we believe the (ME2) is additionally unique for its attention to the process security requirements [3] of embedded computers. With the annual maintenance and development costs of DoD embedded computer resources expected to exceed \$ 40 billion by 1990 [3], now is the time to understand and enforce the security requirements that are unique to these computers. Security in these systems is more than protecting files from users. In the (ME2) development, we have begun to address these requirements. In referring to these requirements, we have used the same terminology as the Naval Research Lab, process security [3].

Currently, we have early prototypes running in our lab. Verification activities [6], and coordination with the NCSC on the development of evaluation criteria for embedded and tactical systems or an evaluation as a TNI network component are the current and short range activities.

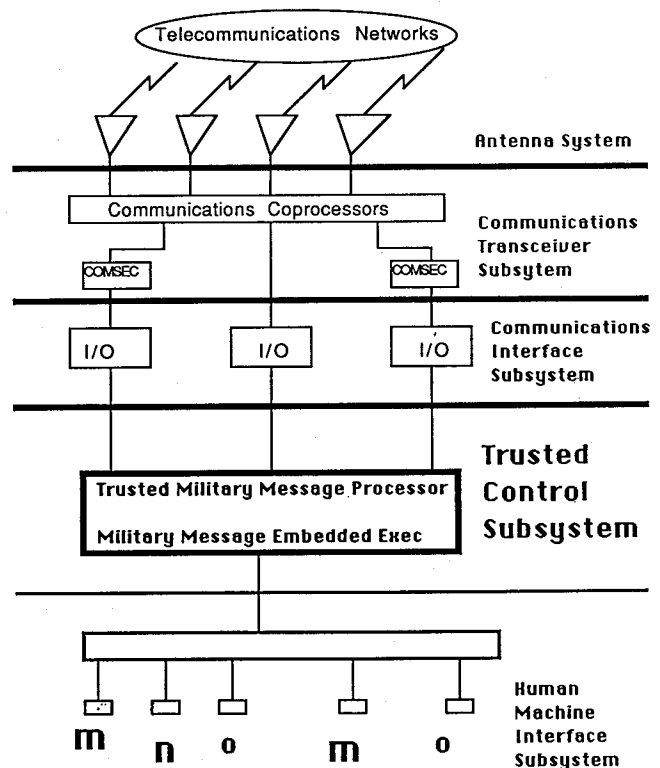


Figure 2. The TRUMMP Environment

REFERENCES

- [1] Landwehr, C. E., Heitmeyer, C. L., and McLean, J., Naval Research Laboratory. "A Security Model for Military Message Systems," ACM Transactions on Computer Systems, Vol. 2, No. 3, August 1984, pages 198-222.
- [2] Landwehr, C. E., Naval Research Laboratory. "The Best Available Technologies for Computer Security," IEEE Computer, July 1983, pages 86-100.
- [3] Froscher, J. N., and Carroll, J. M., Naval Research Laboratory. "Security Requirements of Navy Embedded Computers," Technical Memorandum, 17 April 1984.
- [4] Heitmeyer, C. L., and Wilson, S. H., "Military Message Systems: Current Status and Future Directions," IEEE Transactions on Communications, Vol 28, No. 9, pp. 1645-1654, Sept. 1980.
- [5] Snow, D. W., "State Machine Architectures: An Alternative to Security Kernels for Embedded Systems", Magnavox Electronic Systems Company, 1985.
- [6] Smith, B.; Lindsay, K.; Reese, C.; and Crane, B., Magnavox Technical Report TR MX-NVSG-605/SSE.8801, "A Description of a Formal Verification and Validation (FVV) Process", February 1988.
- [7] "DoD Trusted Computer System Evaluation Criteria," DoD 5200.28-STD. National Computer Security Center, December 1985.
- [8] "Trusted Network Interpretation of the Trusted Computer System Evaluation Criteria," NCSC-TG-005, National Computer Security Center, July 1987.