Bolt Beranek and Newman Inc.

LEVEL II

Report No. 4121

HERMES Security Design

James R. Miller

March 1979

Prepared for:
Naval Research Laboratory and
Defense Advanced Research Projects Agency

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER BBN Report No. 4121	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) 6 HERMES SECURITY DESIGN.		5. TYPE OF REPORT & PERIOD COVERED 9 Technical Report,
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) 10 James R. Miller	12 37 p.	8. CONTRACT OR GRANT NUMBER(s) 15 MDA903-76-C-0212, ARPA Order-3161
9. PERFORMING ORGANIZATION NAME AND ADDRESS Bolt Beranek and Newman, Inc. 060100 50 Moulton Street Cambridge, Massachusetts 02138		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Advanced Research Projects Agency Information Processing Techniques Office 1400 Wilson Blvd., Arlington VA 22209		12. REPORT DATE 11 March 1979
		13. NUMBER OF PAGES 31
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) 14 BBN-4121		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this document is unlimited. It may be released to the Clearinghouse; Department of Commerce for sale to the general public.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

This research was supported by the Defense Advanced Research Projects Agency under ARPA Order No. 3161.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Hermes
Tenex Security
Message Processing

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

In this paper, we describe two successive attempts to develop a version of the Hermes Message System that represent a workable compromise between the goals of security policy, the fact that computer software cannot in general be trusted (or proven correct) and the need for good human factors in an interactive system. Our conclusions are that acceptable human factors must be designed into the system. Intensive efforts should be made to develop effective software verification techniques or other means for making it possible (over)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE 060 100

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. to trust as much of the software as possible.

A

BBN Report No. 4121

HERMES SECURITY DESIGN

James R. Miller

March 1979

Prepared for:
Naval Research Laboratory and
Defense Advanced Research Projects Agency

TABLE OF CONTENTS

TABLE OF CONTENTS (CONTINUED)

LIST OF FIGURES

1. Security Policies for Computer Systems

1.1 Single Level Computer Systems

Today, most commonly used computer systems in the Department of Defense handle data at one security level at a time. There are strict rules that regulate the "cleansing" of a system before the introduction of data at another security level. For example, in order to be sure that no Top Secret data can be read from memory by an unclassified program, core memory must be cycled from all zeroes to all ones at least 999 times before Unclassified material can be processed on the same computer system which has processed Top Secret data.

This policy allows a user access to data at only a single level at a time. Such a policy is acceptable, for example, when a user is analyzing fleet movements and all data is classified at one level. However, in a typical message system the data base consists of messages which are classified at many different security levels, and a user must be aware and able to deal with them at the appropriate level. For example, it may be necessary to reply to an incoming confidential message with either a Top Secret message, or an Unclassified one.

1.2 The MITRE Model as a Basis for a Secure Computer System

In order for a computer system to allow simultaneous access to data of different security classifications, a different approach than that used in single-level computer systems is required. One such approach uses the "MITRE Model" of computer system security reference. This model divides all entities in the computer system into two categories: "Subjects", the active entities, and "Objects", the entities that a subject may access or modify. (Notice that some subjects, such as processes, may also be objects.) The model assigns a security level to each subject and object in the system and defines two major rules relative to them:

1. The Simple Security Rule

   The first rule, called the Simple Security Rule, states that a subject cannot be aware of the existence of an object classified above the subject's security level. This corresponds to the ordinary paper-world constraint that a person without the necessary clearance cannot see a classified document.

2. The *-Property Rule

   The second rule, called the *-property, states that a subject cannot in any way modify an object below its security level. The analog of this rule in the paper-world is less obvious. It is this rule which prevents a program (one form of subject) from downgrading classified material. In the paper world, this rule is tempered by human judgment. Human beings are permitted to read a classified document to which they have access and downgrade parts or all of the document when they deem it appropriate. (Their judgment may later be evaluated by other authorities.) A computer system, however, is not given this freedom, and is prohibited from downgrading any material.

The MITRE Model security policy is considerably more flexible than the single-level model. It allows the computer to store data at several different levels. Thus a user can access data at any level to which he is cleared. Moreover, the user can ask the computer to upgrade the classification of material. For example, he may request that the computer automatically insert Unclassified paragraphs into classified documents.

For the message system application, there is a significant problem with the MITRE model. Unfortunately, there is no way to downgrade data. For example, once a secret document is prepared, there is no way for a human to look at the document and request that a paragraph be extracted from it and stored as Unclassified. (This would violate the *-property.) The only recourse is for the user to request a printout of the document, and then retype the paragraph.

1.3 The 'Security Kernel' Concept

In order to enforce the two rules of the MITRE Model it has been suggested that a computer program, known as a "Security Kernel", be constructed to monitor the operation of other programs. This Kernel must be examined carefully to ensure that it works correctly. In fact, the entire security of a computer system employing a kernel is dependent on a detailed analysis of the security properties of the kernel. Several kernels have been developed to date, although as yet none of them have been completely analyzed; at the same time, a number of "program verification" systems, and "verifiable programming languages" have been developed, aimed at proving these programs to be correct. Although this is still an area of active research, at least two operating systems are being constructed commercially which contain partially verified kernels. <*1>

<*1>. Secure communications processor (SCOMP): Overview of contract and Honeywell development effort. Unpublished handout. Honeywell, Tampa, Florida, July, 1978.

Computer program development specifications (type B-5): Department of Defense kernelized secure operating system. Ford Aerospace & Communications Corp., Report No. WDL-TR7811, Palo Alto, California, March, 1978.

1.4 The MME Security Model

During 1975 and 1976, BBN participated in a joint NAVY/DARPA project called the Military Message Experiment, or MME. As part of this project three different message handling systems were designed and built: SIGMA (created by the University of Southern California Information Sciences Institute [ISI]), MSGDMS (created by the Massachusetts Institute of Technology), and BBN's HERMES. All three systems were intended for operational use at the Commander-In-Chief Pacific (CINCPAC) headquarters.

Since CINCPAC handles a very high volume of messages, and much of the traffic is classified, all three systems were required to have a human interface which might eventually be built in a truly secure manner.

After considerable work on solving some major security problems in the three candidate message systems for the MME, all participants in the experiment acknowledged that a viable message system could not be built on a system which rigidly enforced the MITRE model. In particular, message systems have functions such as Reply which require violations of the *-property in order to work.

For example, imagine a user attempting to reply to a SECRET message with an UNCLASSIFIED response. In order to make a reply, the user may wish to copy part of the incoming message (for example, the subject field which is not stored separately in AUTODIN messages). In order to do this, however, the text of the subject must be formally downgraded from SECRET to UNCLASSIFIED. This is not permitted by the MITRE model, so the user would have to type the text in from scratch.

In order to accommodate these functions with minimum impact on the MITRE security model, a modification to enforcement of the *-property was adopted for the MME:

> The security kernel of the computer system permits a violation of the *-property provided that the kernel receives explicit confirmation of the violation from the human user.

The MME policy requires that the user be aware of the reason for the kernel's confirmation request, and the kernel must always display to the user exactly what data is being declassified. For example, when a user declassifies a document using a display terminal, each screenful of material must be individually confirmed to be certain that all the data displayed is to be downgraded.

The MME policy is a closer model of the security policy in a manual system. It allows a person the ability to examine data at several different levels, and create new information at some other, possibly lower, level based on that data. In a manual system this is not a problem, since the tools available (such as paper, pens, etc.) are completely under the control of the person using them. A computer is different: we probably can never expect to be able to prove that the computer is doing exactly what a user expects (although there is some hope that we can show it does what he commands -- not at all the same thing!). Therefore, there must be some way of assuring that even if the computer is not doing what the user expects, it is at least not violating a security restriction without the user being aware. This is precisely what the MME policy permits: Whenever the computer violates the policy, the security kernel appeals to the user to approve the action that his highly complicated tool is taking.

The three candidate systems for the MME each defined a special part of their system known as a "Trusted Job." These jobs, called TJs, were the only part of the system permitted to violate the *-property. They are extensions to the security kernel used to provide a more elaborate policy decision than those included in the

MITRE model. The TJs are called by the message systems when a *-property violation must occur. The TJ then explains the nature of the violation to the user and asks him to confirm the action. If the user approves, the violation occurs. If not, it is rejected.

The TJ, as described above, is only a sample of the kind of security policy decisions that can be made in this way. Other decisions can also be implemented that allow other kernel enforced rules to be violated. However, the TJ must be proven to operate correctly, since it functions as an extension to the kernel. Also, if it appeals to the user for permission to violate a policy rule, it must "talk the user's language". Therefore the TJs for each system were vastly different -- they were expected to explain each different kind of violation to the user and display the data being downgraded in a manner appropriate to the system and the command causing the downgrade.

By adopting the MME Security model security policy, each of the three systems in the MME competition was able to provide the functions required for use at CINCPAC. Within the security policy, however, each system chose to implement a different user interface and therefore made a different set of design decisions.

2. The Evolution of Secure Hermes

Below, we describe the decisions affecting the security design of the Hermes message system. <*2> Initial design decisions were made during the summer of 1976 in preparation for a formal evaluation period scheduled for March, 1977. These decisions were reconsidered in the fall of 1977. Because this re-evaluation period occurred after the formal evaluation of the three systems, the current security design of Hermes takes advantage of important features in each of the three message systems. We are fortunate in having had the opportunity to do this redesign, and feel that significant knowledge can be gained by studying both the original design and the decision process leading to the more recent design.

2.1 The Hermes System Before the Security Design

Before describing the initial security design of Hermes of March, 1977, we will review some important facts about the original Hermes. Work on the Hermes message system began in mid-1974. In fact, a version of Hermes was already widely used on the ARPANET prior to the design of the Hermes security interface. Since the message system was already quite large, and had undergone several years of intensive review and improvement, we felt that it was essential that the "secure Hermes" be as similar to the ARPANET version as practical. As a result, a deliberate constraint on the security design was that the actual implementation would be added to existing Hermes code under a compile-time switch. This requirement was retained in the fall of 1977 during the subsequent redesign of the security interface.

A number of very powerful reasons caused us to make the secure version of Hermes as similar to the non-secure version as possible.

<*2>. Burchfiel, J. and Myer, T. Message technology research and development. Bolt Beranek and Newman Inc., Report No. 3783, Cambridge, Mass., July, 1978.

For one thing, unlike other military message systems, the systems developed for the MME were intended as experimental vehicles. We were not constrained to handle only AUTODIN formal military messages, but were required to test the use of a computer for processing informal messages as well as internal formal memoranda. In order to accomplish this, it was important to produce a system that was as flexible as possible. We planned to allow the users of our system to modify much of the system as they felt necessary. For this purpose, Hermes already contained a powerful set of tools which had been carefully tested by untrained users on the ARPANET.

Since we were already responsible for maintaining, testing, and distributing new versions of the Hermes system on the ARPANET, we felt that debugging the secure system would be simpler if most of the code was being used daily by our large user community. Thus, if the code could differ only slightly from the standard ARPANET version of Hermes, the incremental cost of maintaining a secure version could be kept quite low.

Finally, by making the underlying data structures for the secure and non-secure versions similar, the secure system could be tested experimentally on the ARPANET, and we could ask selected users of ARPANET Hermes to use the secure version for their daily message processing. In this way, we could actually get feedback from a very large user community even before the system would be available to users at CINCPAC.

2.2 The Preliminary Security Design Using the AIM Kernel

When we first faced the problem of creating a version of Hermes with a "secure human interface," we expected to use a secure version of the TENEX monitor, known as AIM. <*3> Our initial

<*3>. Ames, S.R., Jr., and Plummer, W.W., TENEX security enhancements. Mitre Technical Report No. MTR-3217, Bedford, Mass., April, 1976.

design assumed that we would build a certifiably secure system for which AIM would provide file-handling capabilities.

AIM, which stands for Access Isolation Mechanism, is a version of the TENEX monitor with enhancements necessary to provide a secure system. The enhancements were modelled on the secure MULTICS monitor in use at the Air Force Data Services Center. There were two major enhancements provided by AIM. The first was to add to the TENEX filenames a new field indicating the security level of the file. This security level is used by AIM to enforce the MITRE model rules for file access. The second enhancement is to the interprocess communication facilities to enforce the rules for process-to-process communication. AIM was intended to provide a reliably secure TENEX system, but was not expected to be proven. It was therefore not truly written as a kernel although it can be regarded as an unprovable kernel from a TENEX programmer's point of view.

Under this approach the user views the system as four complete and independent copies of the standard message processing program, one at each of the four security levels. This is an easy model to understand and allows the user to control his security level at all times. Some functions (such as reply) require interaction between the different copies of the program. For example, when the user creates a reply to an incoming message, the subject and references may be copied into the outgoing message; this may involve different security levels or changes in security level. However, this is also easily understood.

This system provides the user with a simple conceptual model that closely matches the actual computer program. In addition, it can be built easily on top of the AIM version of TENEX.

The AIM monitor provides access mediation to all files and separate names for files at different security levels. This gives the user maximum flexibility by allowing the existence of all the Hermes objects at any level and providing some naming convention to allow for unique identification of objects at different security levels (for example, by adding ".T" to the names of Top Secret objects, ".S" to the names of Secret objects, and so on). The only difficulty we foresaw was that when the user displayed a message, only those portions at or below the security level of the current Hermes would be seen. This might lead to some surprises if the user, for example, went to the Unclassified Hermes to display a message and saw only the addressees and the date (i.e., the Unclassified portions), but not the (classified) text or subject. On the other hand, if a user were constantly conscious of his current security level, this would not come as a surprise at all; it would, in fact, be "proof" that the system was working correctly.

2.2.1 The Concept of the Trusted Job

The user logs into Hermes at either the user's maximum security level (i.e., his clearance) or the maximum security level of the terminal, whichever is lower. Commands within Hermes allow the user to change his security level. Hermes includes a "Trusted Job" which acts as the coordinator between the Hermes activities at different security levels.

The Trusted Job, in turn, creates a separate and complete Hermes job at the user's maximum security level and at each security level below. Each "security-level Hermes" job operates independently.

All communication from higher to lower security levels is done through the Trusted Job. This communication is handled as follows:

For any information other than commands without arguments, the Trusted Job first displays the information to the user, requests confirmation, and upon receiving confirmation passes the information to the lower security level. The details of this communication process are:

a. Commands without arguments whose effects can be seen at lower levels are redisplayed to the user and then passed in canonical form. No confirmation is required. This alerts a user that a *-property violation is occurring, and allows him to easily detect incorrect or excessive violations; it also acts to limit the speed with which violations can occur, and hence limits the bandwidth of possible security compromise channels.

b. Commands with arguments which affect lower levels are passed in canonical form and require user confirmation. These are: add, compose (if a template is given), Delete, Erase, Explode, File, Move, Get (if a message-file is given), Release, Reply, Refile, Send and Undelete.

   1. File names are passed to lower levels in the form <DIRECTORY>FILENAME. The directory is omitted if it is the connected directory.

   2. Object names are passed as strings of alphanumeric characters.

   3. Sequences are passed as a list of message numbers; the list may include ranges.

   4. Numbers are passed as numbers.

2.2.2 Message Structure and Message-Files

To the user, a 'message-file' looks like a single file, containing messages with fields at different security levels. In the software, each message-file actually consists of four TENEX message-files, one at each of the four security levels. Each file at a given security level contains only those portions of the messages which are classified at that security level. Each security level file contains a machine-readable header line for every message. Therefore, in order to display the contents of a message, Hermes must examine the contents of all the images of the message in the security level files which are at or below the current security level. Each security level file has its own parse file.

The design is not dependent upon policy decisions about the classification of various message fields, and many different arrangements can be supported. It is possible to have multiple instances of certain message fields at different security levels. This permits, for example, a message to have two different subjects, one unclassified and one secret. While this facility is not currently available through AUTODIN, it can clearly be useful and in fact exists in the paper world: some classified documents are available in unclassified form with the only difference being the name!

The concept of a "current" or "active" message-file is supported. At any time, a copy of Hermes at a given security level can be attached to a single (user visible) message file. The Hermes job will "see" only as much of the total file as the security level permits. Thus a Hermes job running at Secret will have access to the single-level files at Secret, Confidential, and Unclassified and display to the user only data at those security levels.

2.2.3 Output Operations

Output commands are applied to all of the security levels that are accessible at the user's current security level. Message fields above that level do not appear in the output.

Since a machine-readable header exists at all security levels for each message, the presence of a message is known at all security levels. The message numbers, at least, can always be printed out by a user at any level. In practice, fields such as the addressee fields and the date are always required to be unclassified in order to allow message delivery software to be coordinated by an unclassified process.

2.2.4 The Draft Message

HERMES has an object known as "CDRAFT", the current draft message. A user can add, remove, or change fields in the draft message. He can insert text from draft-files into fields and store fields into draft-files. The draft is normally intended for eventual release as a message to other message system users, either within the TENEX system or on another site on the ARPANET or AUTODIN.

The draft message is maintained much the same as a message file. It is composed of a multi-level virtual file, with each constituent physical file containing message-fields that were input through the Hermes job at the file's security level. The draft message is accessible for display (and output) according to the logic described for message-files. A copy of Hermes at a given security level can access all parts of the draft that are at the same level or below. All of the input and modification of the draft fields is accomplished through Hermes at the appropriate level.

2.2.5 The Object Editors

The non-secure version of Hermes consists of a main command level and several subsystems, referred to as "object editors". The objects visible to a user are messages, files, a draft message, templates, filters, switches, user-field dictionary, and sequences. In designing the security interface to Hermes, it was necessary to make decisions about the classifications of each of these objects. These decisions have an impact on the user interface to the editors. For example, if a template (which is a pattern for creating or displaying a message) can be used to store classified data (such as a standard subject), it must be classified, and the template editor must be able to handle templates at different security levels.

2.3 The Change to a Simulated Security Design

Later in the development of the secure Hermes system, the assumption that AIM would be present was eliminated. We were therefore faced with the problem of designing and implementing a system without an available security kernel. The solution for the MME experiment was to design and build a system with simulated security, in the expectation that we would adapt it to a security kernel when one became available.

At this point, a major difficulty arose with our model based upon four complete copies of Hermes. Any such system requires large amounts of "security bookkeeping" to keep track of different versions of different objects at different security levels. Since we did not have the AIM security kernel to perform the security bookkeeping, this would constitute a major performance burden on Hermes.

2.4 The First Secure Hermes -- March 1977

As a result, we re-evaluated our design, and decided that although we would keep the quadruplicate structure of the message-files, we did not need to operate four simultaneous Hermes systems for message display. We changed the design of the top-level commands so that instead of giving the user a choice of security levels for reading messages, the system always operates at the User's maximum security level.

2.4.1 Message Display

Our initial goal for the secure version of Hermes was that it should have the same set of commands, command syntax, and command input style as the non-secure version. This requirement precluded the use of techniques which made active use of the good features of the video terminals which were provided for the experiment. For example, we believed it was more important to have the advantage of an acceptable, tested, user interface. With this in mind, we examined the existing commands to see where security issues would require that change be made.

The most obvious impact is in displaying the message. We felt that the flexible message structure in use on the ARPANET, consisting of arbitrary headers and text, should be maintained in the secure version. Further, it was clear that some headers (such as the date the message was created) were unclassified, while others (such as the subject) had to be classified. Since our goal was to make Hermes flexible, there were no constraints placed on the classifications of these header fields. Instead, we said that there would have to be three categories of header fields: a) those restricted to a particular security level (either by other factors in our design, or by administrative fiat), b) those restricted to a single security level selected by the creator of the message, and c) those which could contain data of more than one security level.

2.4.2 Access to Message-Files and Messages

Having settled on the security aspects of the internal structure of a message, we began to concentrate on that of the overall message and message files. While several people involved with the project advocated a policy of "message classification" to restrict access to all parts of a message, we felt that this was unnecessary. We interpreted the security regulations to allow anyone who had access to the message-file containing a message to see the parts of the message for which he was cleared. Further, mostly for ease of implementation, we did not provide for minimum access to message files either. Hence access to a message was dependent on privacy controls <*4> provided by the operating system (and not necessarily guaranteed by the security kernel), while the security rules were enforced on individual fields of the message.

2.4.3 The Creation of the Draft Message

Our next step was to consider the draft message. Since non-secure Hermes makes no distinction between messages which have been transmitted with the SEND command and those which have been placed in a message-file by some other command (such as File or Redistribute), we felt that the same should be true of the secure version. Therefore, we construct all messages as if they were for internal use in Hermes. Different fields may be protected as single level or multi level objects. Each header field is either fixed at a system-defined level, a user-defined level, or it can contain portions at several levels. Messages released to AUTODIN are converted to single-level objects at the time of release and not before. This permits the Hermes user to have great flexibility when dealing with messages within the TENEX environment. It is only when a message is to be moved outside of TENEX that more restrictive regulations are imposed.

2.4.4 Draft-Files (files containing text)

Files that contain pieces of text intended to be used within the Draft-Editor are called "draft-files" in the Hermes system. Non-secure Hermes has two classes of draft-files: (1) unstructured

<*4>. The term "privacy control" is used here to mean access controls imposed outside the kernel. On TENEX these controls are based on the login name of the user requesting the access.

text files, which can be used to store single fields of the draft message. These are handled with the Store-Field and Append-File commands. (2) Structured files, which contain more than one field, but which are not in a form to be accessed by Hermes message-reading commands. These are handled with the Store-Draft and Restore-Draft commands. We did not use the structured draft-files in secure Hermes since the Edit Message and Refile commands allow unsent messages to be saved in message-files and handled with the same tools as messages.

2.4.5 File-Names

One of the functions of the security kernel in the initial security design was to supply file-names that contain security information to Hermes. In order to provide this function without the AIM kernel, we implemented the Hermes file-name mode of operation. Under this scheme, the "extension" portion of the TENEX file-name <*5> is used to tell Hermes the distinction between draft-files, which are classified at a single level, and message-files, which are multi-level. The file-names appear to the user to be single words consisting of the first portion of the TENEX name. For example, the file "MESSAGE.TXT;1" appears to the user as "MESSAGE". All files with the extension ".TXT" are recognized by Hermes as multi-level message-files. The single-level draft-files are given the extension U, C, S or T. depending upon their classification level, for example, PEOPLE.U;1 or WARNING.S;1.

<*5>. Myer, Theodore H., Barnaby, John R., and Plummer, William W. TENEX executive language manual for users (Revised edition). Bolt Beranek and Newman Inc., Cambridge, Mass., January, 1971, revisions published April, 1973.

2.5 Hermes Objects Other Than the Draft Message

Although the Hermes objects known as sequences, filters and templates could all contain classified information or be given classified names, we made the decision to require them to be unclassified in Secure Hermes. <*6>

This decision was made because of time constraints. Under the original design, if the facilities of the security kernel are not available for handling object names, some means would have to be found to prevent duplication of object names at different security

<*6>. The classification of an object and the name of an object are very different concepts, and have different effects. Consider three cases: the object and its name classified at the same level, the object at a classification above its name, and with the name above the object.

The first case is the 'normal' case. The object is accessible to anyone who has access to its name -- notice that this statement is not true of access to files on most traditional computer systems, where access regulation rarely applies to names of files.

The second case, with the name classified below the object, leads to a situation where a subject can appear to be aware of the existence of an object classified above its level. In fact, the subject is aware only of the NAME of the object -- something which is not classified above the subject's level. This is similar to the situation discussed above where the message number is known at all levels within HERMES, even though there may be no fields in the message.

The third case, with the name classified above the object, is frequently ignored as "ridiculous". There are, however, at least two situations where this can be useful. It allows for one form of minimum access control, since most references to an object will come through the object's name. If the system allows other (internal) references to the object, however, then it is possible to implement another scheme. As mentioned above, an object could have several names, each at a different security level. If instead of "name" we think of a "pointer", then this allows for a piece of unclassified text to be contained in a secret document (via a pointer) and yet still be accessed from outside of that document as unclassified text. Depending upon the security policy that is to be enforced on overall classification this can be either a distinct advantage or a major objection to this form of a pointer implementation.

levels, without passing information about these names from a higher to a lower level.

2.6 The Design Consequences of the Trusted Job and the -property

Once the general design of secure Hermes was established, consideration of the trusted job and the *-property revealed further problems. These are described below.

2.7 Object Editors

Although the secure Hermes objects such as sequences and filters are themselves unclassified, the commands used with the editors may involve potential violations of the *-property. For example, in order to add all messages with "ROSES" in the subject field to a sequence of messages, the user gives the command:

```
>>Add Subject: Roses
```

to the sequence editor. The search must be performed at maximum security level because a subject field may be at any security level, and the word being searched for may itself be classified. Since the resulting sequence will (like all Hermes objects) be unclassified, security requires that when the search is completed the results of the search must be redisplayed to the user. The user must confirm the downgrade of the sequence from its current level, that of the search, to the eventual level of the sequence (unclassified).

Example:

```
User types: >>Add Subject: Roses

TJ types:   Add 1:5,7,15             [CONFIRM]
```

2.8 Terminal Security

Since each message is displayed at the user's maximum security level, several different security levels may be shown on the screen at any one time, and it is necessary to keep the user informed about the maximum security level on the screen.

Our solution to this problem is to provide a vertical security bar at the left margin of the display screen, which shows a security level for each line of the display in reverse video. The maximum level on the screen is displayed in a summary across the top.

3. The results of Experience with the First Security Design

We asked a number of military and civilian users, many of whom had never had experience with computers before, to experiment with the first secure Hermes. As a result of this work, we found three major problem areas in the design of the user interface.

3.1 The Confirmation Problem

The numerous confirmations required by the security design proved to be very annoying. They were too frequent, because most commands were typed at the user's maximum level, and had to be downgraded before each command was processed. In addition, to guarantee to the security kernel that the user understood that he was confirming a downgrade and hence a possible security violation, the user was required to use a special key not normally used by Hermes. The result was that the users perceived the confirmations to be dissimilar to normal Hermes commands.

3.2 The Problem of Changes in Security Level

The users felt that the concept of being "at a security level" while drafting a message was very confusing. Contrary to our expectations, the users expected the computer to move them to the correct security level for the part of the job at hand, rather than having the users instruct the computer to change levels. The users clearly would have preferred a less flexible message form if it relieved them from the burden of thinking about the security level of each part of the message.

We analyzed the choices that could be made for assigning security levels to message fields. A tree-structured diagram of these choices is shown in Figure 1. The design of the first secure Hermes occupies one position on the tree, and the two alternative methods of using the second secure Hermes occupy two other positions.

Figure 1.

3.3 The Problem of Interruptions by the Trusted Job

There appeared to be too many interruptions by the Trusted Job to announce changes of security level.

4. The Second Security Design -- November 1977

In November 1977, ARPA requested that BBN resume the CINCPAC related work that had been suspended in March 1977, and that we prepare Hermes for possible use in the MME.

4.1 The Reactivation of Secure Hermes

We reactivated and installed the CINCPAC version of Hermes in our research computer center, and we obtained and reviewed documentation on the evaluation of Hermes from the preceding March. After further consultation with potential Hermes users. We undertook modification and extension of the secure Hermes software.

To provide a simplified command repertoire and improved performance, we converted to our H2 software (which presents the user with a subset of Hermes commands) as a basis for CINCPAC support. We then designed and implemented an entirely new security interface with improved security safeguards and substantially improved human factors.

4.2 The Revised Security Design for Hermes

The user interface associated with the second version of secure Hermes provides a simpler system, with a smoother transition between security levels than the system demonstrated in March 1977.

Our goal was to ensure that operations take place at the appropriate security level, and that, as far as possible, changes in security level occur automatically. The only times that the user is required to supply security level information is when he reclassifies a message or creates a new message.

4.2.1 One Security Level per Message

Each message has only one security level. Fields that identify originators and recipients, dates associated with the message, the classification of the message, the message type, and the precedence, are required to be UNCLASSIFIED. Other fields may be classified but they are all classified at a single level within a given message.

4.2.2 Automatic Transitions Between Security Levels

Each user logs in at his maximum security level and then is automatically transferred to UNCLASSIFIED. The user is at the UNCLASSIFIED level whenever he inputs commands. If the commands call for display of a message, the user is automatically transferred to the security level of the message so that all the information is displayed. A user cannot display messages classified above his maximum level.

If the user wishes to set a lower maximum security level for the current Hermes session, he can give the command

```
>MAXIMUM SECURITY LEVEL <classification><CR>
```

This security level will remain in effect until the user logs out.

4.2.3 Changing Security Levels

When the user gives a command for message composition, such as COMPOSE, the series of prompts presented to the user is controlled by a Hermes template. One type of template item automatically asks for the security level of the message. As the user responds to the series of COMPOSE prompts, the message is automatically created with fields like To:, Cc: and Date: at the required UNCLASSIFIED level, and fields such as Subject: and Text: at the classification level that has been specified by the user. When the user SHOWS the unsent draft message, Hermes also makes the transition between security levels, without efforts on the user's part.

If the user desires to change the classification of the message, he may do so at the end of the COMPOSE command. The user is left in the draft-editor where he can use the RECLASSIFY command before sending the message. The user may also add more fields, edit or erase existing fields, show the draft message or Send it, as in regular Hermes.

When the classification is upgraded, only the RECLASSIFY command is necessary. If the RECLASSIFY command is used to downgrade, Hermes displays the entire draft for review and requires the user to confirm the reclassification.

The user is continually informed of the classification of each line displayed on the terminal by means of an inverse video "bar" displayed at the left-hand edge of the scope screen. The first four character positions are used to display the classification character (U, C, S or T) that applies to the line.

Whenever the classification at which the user operates is changed, the user is notified by an inverse video line, e.g.,

    FROM UNCLASSIFIED TO SECRET

Similarly, any confirmation of change of classification required by the system is highlighted by inverse video.

4.2.4 Multi-level Fields Removed

Multi-level draft fields were abolished in accordance with the new security concept of automatically moving the user to the correct security level. If it is possible to create a field at more than one level, it is not possible for the computer to determine the "correct" level, so the user would have to supply the information.

4.2.5 Security Downgrading Confirmation is Rarely Required

1. The user is required to reconfirm when leaving the sequence editor, since editing that involves message-fields always involves a potential security violation. There are two opinions about requiring confirmation. (a) the user could be required to confirm every time he leaves the sequence editor, even though the commands that he gave involved only fields required to be unclassified. The user would then not be surprised by an unexpected requirement for confirmation. (b) The user should be asked for confirmation only if the commands involved fields that might be classified. Surprise is desirable because it calls the user's attention to the fact that there might be a security violation.

2. Confirmation is required when the user gives a command that requires Hermes to look at fields at the maximum security level. This occurs when the message specification for command involves a potentially classified field, such as the subject field. Commands which can be given such a message specification are Redistribute, Append, Assign, Comment, Explode, Remove, Add, JumpTo, File, Move Consider, Delete, Undelete, Mark or Reply.

3. Confirmation is required when the user sorts a sequence on a potentially classified field.

4.3 Restricted Access to the TENEX Executive System

Security is enforced by restricting the access of Hermes users to the TENEX Executive System. <*7>

<*7>. Myer, Theodore H., Barnaby, John R., and Plummer, William W. TENEX executive language manual for users (Revised edition). Bolt Beranek and Newman Inc., Cambridge, Mass., January, 1971, revisions published April, 1973.

Users are placed directly in Hermes upon logging into the system, and must logout directly from Hermes. There is no provision for dropping into the Executive system in a lower fork through the EXEC command as there is in regular Hermes.

5. Summary and Conclusion

The need for military security in computer systems generates a three way conflict between the goals of security policy, the fact that computer software cannot in general be trusted (or proven correct), and the need for good human factors in an interactive system. Computer security research can be viewed as a continuing search for acceptable compromises between these three fctors.

Conflict, and the need for compromise, are present for the following reasons.

1.) Basic security policy governs human behavior in a paper world. This policy works well because the paper world is simple, and because the humans who have been given access to classified information are trusted to adhere to the rules with regard to that material. For example:

- Multiple security levels may occupy a single sheet of paper (or multi page document).
- Provided the environment is secure, the user may freely write information at any level or intermixed levels. The decision on how to classify the results of his writing can be left till the writing is complete.

2.) Within the computer, things are neither so simple, nor can the computer and its software taken together be trusted to adhere to the security rules. In particular:

- Information must be segregated by security level, even though that information may constitute a single document or message.
- Information, including commands, must be entered into the system at its correct level. To enter information at too low a level would leave it exposed to unauthorized access. Entered at too high a lvel, the information would require subsequent downgrade, which cannot be entrusted to computer software.

3.) The goal of good human factors implies ease of use. In terms of security, a computer system with good human fctors would at the very least be as straightforward as the paper world. Unfortunately, this goal is in direct conflict with point 2 above, which tends to imply a system of considerable complexity.

In this paper we have described two successive attempts to develop a workable compromise between these forces. Like many such efforts this one began with relaxation of the restrictions placed on computer software. In particular, the ability to downgrade information, but with human confirmation of each such downgrade proved to be a necessary relaxation of the original MITRE model. To make possible this extension, the Security Kernel of trusted software that is generally created to support the MITRE model was augmented by a "Trusted Job" that carries out user assisted downgrades.

Starting from this point of departure we proceeded to create two successive security designs, the second more acceptable than the first in terms of human factors.

The first design took a "do it yourself" approach. It assumed that the user would be willing to manipulate directly the rather intricate structure through which the security rules were to be implemented in the computer. This design was modular, and, we thought, elegant. It allowed the user to move freely about the security levels viewing what the security rules would allow at that level. Messages could contain any mix of classified information, and thus a message might appear quite different at each viewing level. Draft messages were constructed by shifting to each of the security levels to be contained in the draft, and then entering the information classified at that level.

In a second stage of this initial design, we modified the system so that all viewing operations were carried out at the user's top level, so that the maximum possible information could be seen. However, draft composition was still carried out by stripping the system through successive levels.

Neither version of this first design proved satisfactory to our users. What we thought straightforward and modular proved arcane and confusing. It became clear that users would prefer to have the system shift levels automatically as required, without confirmations and without the numerous notifications required by successive level changes.

In the second design, we restricted the number of security levels per message to just two (unclassified and one other); we provided for command input at the unclassified level (which avoids the command downgrade problem) and we arranged for almost all level changes to take place automatically. Only when indicating the security level for a new message or when reclassifying stored information must the user be aware of security level. This second design proved much more satisfactory to our users.

This research emphasized the human use of a multi level secure system. The dominant conclusion is that for such use to take place at all, acceptable human factors must be designed into the system.

Good human fctors would not be hard to achieve if the system software could be trusted not to violate basic security provisions (especially writedown). It is our present inability to verify and therefore trust software that makes it such a struggle to achieve good human factors. Thus, a second conclusion is that intensive efforts should be made to develop effective software verification techniques or other means that would make it possible to trust as much of the software as possible.