Attachment No. 1
24 June 1974 (revised)

STATEMENT OF WORK

Security Kernel Evaluation for Multics

1.0 Introduction

The military has the responsibility for protection of information in its shared computer systems. The military must insure the security of its computer systems before they are put into operational use. That is, the computer system security must be "certified", since once military information is lost it is irretrievable and there are not legal methods for redress.

Most contemporary shared computer systems are not secure because security was not a mandatory requirement of the initial hardware and software design. The military has effective means of implementing and certifying physical, communication, and personnel security, so that the nub of our computer security problem is the certification of information access controls in the operating system and supporting hardware. The primary need is for an effective means for enforcing simplistic protection relationships, by implementing a certifiable "security kernel". Initially, solutions to some of the more complex protection problems such as mutually suspicious processes is not required.

The purpose of this contract is to design and develop the interfaces to the so-called security kernel of a general purpose computer system which provides controlled, direct sharing of information and programs between users with different authorized access. In addition, the results of this contract will define those non-kernel security related functions that will be required in such a computer system.

2.0 Scope

2.1 Objective

The contractor shall review the functional primitives of a security kernel for the Multics general purpose multi-user computer system. These primitives

shall implement the necessary functions to meet the Department of Defense security requirements for the protection of classified information. These functional primitives shall deal only with the security of information objects and their access. Excluded from the security kernel functional primitives are operating system primitives which do not affect information security. Included in the class of excluded primitives are those providing system integrity such as preventing denial of service, page replacement and scheduling strategies, resource allocation policies, and other non-security related operating system functions. In reviewing the security kernel primitives, the contractor shall suggest alterations to ensure efficient interfaces with the non-security related portions of the operating system. A secure Multics would contain certain non-kernel security related functions including but not limited to login verification, input/output daemons, and security officer functions. The contractor shall define these functions and identify special problems posed by their interfaces to the security kernel.

In addition, the contractor shall conduct a study of the integration requirements for the design and implementation of security software. This study shall identify the scope and detailed integration tasks for a prototype design and implementation effort.

2. Approach

In examining the operations of the kernel, the contractor shall be guided by the following principles:

a. Complete Mediation -- A secure system must provide complete security mediation of information references. All references must be validated by those portions of the system hardware and software responsible for security.

b. Isolation -- The validation operators, a "security kernel", must be an isolated, tamper-proof component of the system. This kernel must provide a unique, protected identity for each user who generates references, and must protect the reference-validating algorithms.

c. Simplicity -- The security kernel must be simple enough for effective certification. The demonstrably complete logical design should be implemented as a small set of simple primitive operations and system

data base structures that can be shown to be correct.

To ensure correct and consistent operation and to provide a method of certification, the kernel functional primitives shall be defined from an abstract model of computer security which has been mathematically proven correct. Such models are being developed under related contracts and the Government will furnish the products of this development to the contractor. Then if an actual instance of the security kernel can be shown to correctly implement the functions of the model, the kernel can be certified correct, since the model has already been proven correct.

The contractor has previously identified several programming and structural techniques (e.g., the accomplishment of dynamic linking outside the supervisor and the use of a structured programming language) which may help to simplify the Multics operating system. The contractor shall investigate these and other techniques for possible benefits in easing the certification problem or simplifying the kernel structure.

The Multics system architecture will be the target of the integration requirements study. The study will also consider the prototype secure computer system as the context for future system engineering or the design of a secure front-end processor, the requirements and characteristics of surveillance and audit in a secure system, and the economic/technical potential of a secure low cost office terminal.

3.0 General Background

3.1 Deficiencies of Present Systems

Most current computer systems exhibit a complex, ad hoc security design with a diffuse implementation that violates the third principle cited above, simplicity. Large portions of complex operating system execute in an all-powerful supervisor state, so that the entire operating system has potential security implications. Whatever nominal security controls exist in such bug-prone monoliths are not effectively isolated (in violation of the second principle) and so can be tampered with by exploiting errors or trap doors in other parts of the operating system.

The significance of these inherent security weaknesses has been amply and repeatedly demonstrated by

the ease with which contemporary systems have been penetrated. Unfortunately, this lack of an underlying design methodology cannot be effectively overcome by ad hoc "fixes" and "security features". Although the current Multics system has basic security design features, it has no precise security criteria which provides a basis for certification.

3.2 Certification

A naive (but occasionally attempted) approach to insuring the security of a complex operating system is to have a penetration team of "experts" test the system. It is supposed that repeated unsuccessful penetration attempts demonstrate the absence of security "holes". A security evaluation through such attempts may reveal weaknesses of a system but provides no indication of the presence or absence of trap doors or errors in areas unnoticed by the attack team. The failure of an attack team to notice a particular penetration route does not provide a basis for proving or certifying that any future penetration attempt will overlook it. The underlying concern is that an active penetrator is not particularly thwarted by the various flaws found and fixed through testing so long as there remains **just one** vulnerability that he can find and effectively exploit.

On the other hand, the three principles identified above (viz., complete mediation, isolation, and simplicity) should lead to a simple, well-defined subset of the system totally responsible for information protection. The goal is that the primitive functions of this small, simple kernel can be tested by enumeration, and other parts of the system are not relevant to security. As a result of the small portion of the operating system included in the kernel, most system changes will not affect the kernel, so routine system maintenance will not require repeated recertification.

4.0 Contractor Tasks

The contractor shall develop, maintain, and submit a detailed set of technical working notes relating to each of the following specific tasks throughout the course of the project.

4.1 Analysis Activities

In pursuit of the objectives of this contract the contractor shall conduct the following general

activities throughout the course of the project.

4.1.1 Review Security Kernel

The contractor shall perform a detailed review of a set of security kernel functions and primitives to be developed under related efforts to be identified by the Government during the performance of work under this contract. The contractor shall attempt to insure, based on his experience with Multics, that the primitives form a complete set for security, but do not contain unnecessary functions (from the standpoint of security). The contractor shall prepare the results of this review as a technical report identifying unnecessary functions in the kernel design supplied by the Government, functions that should be added to the kernel, and proposed restructuring to improve interfaces between the kernel and the remainder of the operating system.

4.1.2 Define Operating System Interfaces

The contractor shall define a revised organization for the Multics operating system to interface with the security kernel. He shall identify design and structural techniques by which the operating system may simplify the kernel's organization and functions. These techniques include but are not limited to restructuring for parallel processing, replacing bulk memory with large primary memory and using ARPANET techniques for input/output. If the ARPANET interface is used as the front-end I/O processor, then the evaluation of the security kernel for Multics must consider the security implications of the ARPANET.

4.1.3 Definition of Non-Kernel Security Functions

The contractor shall define those non-kernel security related functions (e.g., user password authentication) which are required to be part of the Multics operating system. The definitions of these functions shall include considerations of ease of certification, simple user interface, and efficient interface with security kernel.

4.2 Reduction of Hardcore to Permit Certification

The contractor shall accomplish the following specific tasks during the course of this contract. The following paragraphs describe several specific tasks which so far have been identified as plausible candidates for

the restructuring of Multics. Several of the tasks suggested here involve modifications to the current Multics system. For each of these, two observations are in order: 1) a method of measurement of progress is needed, to establish "how much" each modification carries the project toward the goal of an auditable central core; and 2) discussions and negotiations with the Multics development team are required to establish whether or not each suggested modification should be targeted toward installation in some current or future standard version of Multics. It seems inevitable that at least some of the changes which will be needed to achieve an auditable system will violate either compatability or performance constraints of the standard system, and thereby force development of a parallel version.

Most of the initial tasks are directed toward identifying more exactly which functions of the operating system must be privileged, and which, by careful design, can be left to the user (in Multics, on a per ring basis.) This work may be described as better defining where the security perimeter of the system should be located. It is expected that there will be many more such tasks in this class. Two remaining major areas of work, both more suitably tackled later, are the rewriting of otherwise untouched protected programs in a standard auditable style, and installation of at least one internal firewall or protection ring within the protected supervisor to separate those procedures which actually implement the protection mechanism itself--a so-called "security kernel". The tasks so far identified are the following:

4.2.1 Removal of the dynamic linker and library search modules from ring 0. This modification would remove two large and hard-to-audit modules from the protected area. The dynamic linker is especially hard to audit because its correct operation depends on its interpreting a highly structured but unprotected data base (an object segment linkage and definition area) without accidentally getting mixed up. Neither of the modules has need for supervisor privileges or protection from the invoking user; both are currently in ring 0 because of their intimate interface with the storage system. The task includes better definition of the interface to the storage system, and taking advantage of the lower cost of changing protection rings with the 6180 hardware.

4.2.2 Removal of the "reference name" concept from ring 0. The notion of a remembered reference name is currently maintained on a per-ring basis, in the

per-process known segment table in ring 0. There is no apparent reason why reference names cannot be remembered in the ring of interest; such an arrangement will also permit a subsystem writer to disable reference names if he desires. This change would simplify both the implementation and the description of several supervisor interfaces.

4.2.3 Removal of the "working directory" concept from ring 0. The comments regarding reference names apply to the working directory also.

4.2.4 Develop a uniform storage system status-returning entry. This minor cleanup would replace about half a dozen distinct supervisor interfaces with a single, more easily audited interface for returning to the user any status information about his segments. (This task is actually the iceberg tip of a larger task to develop a simple, consistent set of supervisor entries.)

4.2.5 Modify the traffic controller to provide cheap, rapidly scheduled, wired-down processes which can operate using any descriptor segment which happens to be available in primary memory. This change would allow the present interrupt handlers for the printer, teletype interface, network interface, and tape handlers to be replaced with scheduled processes. The actual interrupts would do nothing but notify the appropriate process. The virtue of this strategy is that scheduled processes can coordinate their activities with standard coordination primitives (block, wakeup, wait and notify); the present interrupt handlers cannot, for example, wait on an interlock, and are therefore filled with tricky code which uses read-alter-rewrite instructions to avoid encountering interlock situations.

4.2.6 Modify the traffic controller (and other per-process data base managers) to permit multiple processes per address space. This modification is the key to untangling several very complex paths through the present supervisor. Typewriter management, network interface management, dialup handling, and quit handling can all be done as simple coordination of parallel processes rather than with the present ad hoc multiplexing of a single process among many conceptually parallel activities. The propagation of this change through the network control is part of the task, to test its effectiveness.

4.2.7 Develop a uniform process coordination/message passing strategy. The current Multics has several different coordination and message passing schemes in it, each with slightly different properties as to the scope of naming and details of interface:

-- Wait and notify, used for storage system signalling
-- Block and wakeup, used for I/O coordination
-- Interprocess communication, used for multiplexing processes among event call channels
-- Signals, used to generate interrupts in a process
-- Message segments, used to queue messages in a catalogued place
-- Mail facility, used for inter-user mail
-- Lock and Unlock, used for coordinating data base use
-- The I/O system, used for message passing and queueing

The task here is to develop one or two moderately flexible process coordination and message passing facilities which can be used to support all of the various users of these facilities. The payoff in simplification of the central supervisor should be quite high.

4.2.8 Merge the network interface with the typewriter communications interface. These two interface programs are two of the largest protected subsystems; they largely duplicate each other. The typewriter control system should use the network code conversion strategy which does not require protection; the network interface should use a buffering strategy more similar to the typewriter modules. With moderate effort, the interface between the 6180 and the DataNet 355 communications computer can be made essentially identical to the network host-to-IMP interface, allowing further control program sharing. By taking the best design from each of the two systems, a compact and effective communication interface module should result, with minimum privileged code.

4.2.9 System Census. This task consists of conducting a census of the number of programs, number of lines of source code, and number of lines of generated text (machine instructions) in the protected supervisor. This census will be useful for two purposes: identifying subsystems which are unreasonably large or complex for further study, and to keep track of progress in simplifying and reducing the size of the protected supervisor.

4.2.10 ALM program catalogue. A list of all protected programs currently written in ALM (the Multics Assembly Language) should be developed, with the goal of identifying all reasons why assembly language has been used. This task includes the development of proposals to eliminate the need for assembly language completely. Such elimination is an important step in simplifying the description of the system and of simplifying the job of an auditor.

4.2.11 Development of coding style standards. A standard programming style will need to be developed, one which emphasizes clarity in program structure to an auditor. Undoubtedly, the programming style will borrow much from the emerging area of structured programming. The task includes the experimental rewriting of some parts of the storage/directory system to the new standards to test their viability.

4.2.12 Use of unique segment numbers. The implication, in terms of simplifying system structure, of using unique identifiers for segment numbers will be explored. An immediate implication of such a strategy would be that pointers containing segment numbers could be left in permanently catalogued, shared storage; many programmed tricks to accomplish the equivalent effect could be eliminated from the system. There are many other implications for system creation, interprocess communication, dynamic linking, and hardware addressing architecture which should be examined; many simplifications seem to follow. An intermediate strategy, of using unique identifiers to replace the absolute addresses in a segment descriptor word, and developing a microprogrammed memory controller architecture which responds to such unique identifiers and contains in a separate box all virtual memory implementation seems worthy of exploration as part of this task.

4.2.13 Reconfiguration hardware proposal. A fair amount of very intricate machine language code in the protected core of Multics is devoted to the dynamic reconfiguration of processors and memory, a valuable feature. Much of the intricacy can be attributed to performing reconfiguration with hardware not designed for it. A general design developed by R. Schell in his 1971 Ph.D. Thesis should be reviewed and a specific hardware proposal for the 6180 system should be constructed along the lines suggested by Schell. Such a design would probably influence future rather than current versions of the Multics hardware but the result is of interest now to

establish how large is the effect in reducing complexity of the protected supervisor. In addition, operation of a secure system probably requires padlocking many of the control panels currently used by the operator to accomplish dynamic reconfiguration.

4.2.14 System description improvement. If an auditor is to review a supervisor program for correctness, he must have a complete, concise statement of what the program is intended to do. Today's description consists of English language supervisor interface descriptions, with PL/I calling sequences. There is no simple description of the "state" of the supervisor and the things a user may do to legally alter its state. The first step in this task is simply to collect in one place all the present documentation of the protected supervisor interface, and evaluate it. The next step is to try to develop a more precise state description of the supervisor, and the ways in which a user can change or observe its state. This task seems to include becoming expert in description languages, such as the Vienna Definition Language, so as to develop equivalently powerful methods of describing an operating system.

4.3 I/O and Modeling Tasks

In addition to the direct efforts to reduce the Multics hardware, the contractor shall conduct the following examinations and investigations.

4.3.1 Secure Input-Output Processing.

This effort will consist of the examination of current Multics practices in the areas of input-output and communications processing with regard to their conformance or nonconformance to the principles of a certifiably secure system. Technology reports and functional specifications shall be produced for secure front-end processing and secure input-output processing. A key result of the thesis just completed by D. Clark at MIT is that with correct design, essentially no I/O strategy or device management code, except that dealing with multiplexed channels needs to be protected. Since I/O software is a significant part of the present protected supervisor, a detailed design proposal for a new hardware I/O architecture along the line of Clark's thesis is in order. Thanks to the modular organization of the 6180, it is relatively easy to envision actually building and trying out this design at some point in the future.

4.3.2 Consistency with Modeling Activities.

Each effort shall be monitored by personnel familar with and qualified in the mathematical investigation of certification now underway under Government sponsorship, in order to ensure consistency of the proposed technology development activities with the results of the mathmatical investigatin.

4.4 Integration Requirements Analysis

The contractor shall determine and evaluate the future work required to design and implement security software on the Multics system. He shall address the overall work to plan and integrate the complete elements of ADP systems security. He shall consider not only the central computer but also the user interface elements needed to make secure computing practical and available. He shall define the planning and integrating tasks on the central computer and its operating system software, a front-end processor with multiplex cryptographic capability, a secure office terminal, and applications engineering such as a secure data management system and security audit and surveillance.

4.5 Review Meetings

The contractor shall conduct bi-weekly review meetings, as required by the project manager, at the contractor's facility. The contractor shall present reports of progress and projected tasks for completion of the effort. Program coordination and technical guidance will be provided by the program manager (ESD/MCI), through the Administrative Contracting Officer.

4.6 Data Management

All contractor data and reports are required as specified in the attached DD Form 1423. The contractor shall forward a copy of each letter of transmittal for data items directly to ESD(MC-1) Data Management Office, L G Hanscom Field, Bedford, Massachusetts 01730, at the time of distribution.

5.0 Government Furnished Support

The Government will furnish interactive, remote access computer programming support on a Multics computer system. It is the responsibility of the contractor to provide the required remote terminals, communication

facilities, and communication lines. The Government shall provide limited permanent on-line storage space for contractor programs, data, and document text files.