AF/ESD

17 January 1973

STATEMENT OF WORK

Abstract Model for Secure Computer Systems

1.0 Introduction

The military has a heavy 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 security must be "certified", since once military information is lost it is irretrievable and there are no 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 reasonably effective physical, communication, and personnel security, so that the nub of our computer security problem is the information access controls in the operating system and supporting hardware. We primarily need an effective means for enforcing simplistic protection relationships, but we do not require solutions to some of the more complex protection problems such as mutually suspicious processes.

The purpose of this contract is the development of an abstract model of the security-related portions of a general purpose computer system which provide controlled, direct sharing of information and programs between users with different authorized access. This model must provide a design guide that is applicable to development of a secure operating system.

2.0 Scope

2.1 Objective:

The contractor shall develop an abstract, mathematical model of the design requirements implicit in the Department of Defense security system for protection of classified information. The contractor shall also develop an explicit methodology for application of this

model to the design of a specific system. The contractor shall demonstrate this methodology by application to a system with functional capabilities like those of the Multics system.

2.2 Approach:

The mathematic model, as applied to any specific system, shall be developed as a set of levels of abstraction, with the top-most level representing a schema and the lowest level representing the hardware. Attachment 1 to this statement of work provides a brief outline of possible levels of abstraction.

The contractor shall begin the development of his model by abstracting a schema from the various mathematical models of access control extant in the field of computer security. These models are based on controlling the access of active subjects to various objects within the system. The methodology for applying the contractor's model must insure that the completeness and correctness of the model are preserved in a specific design, viz., the methodology must include techniques for proving that the resultant design is correct from the security standpoint.

In developing the model and associated design methodology, 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 -- These 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.

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, simplicity. Large portions of complex operating systems 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 through errors or trap doors in other parts of the operating system.

The significance of these inherent security weakness has been amply and repeatedly demonstrated by the ease with which contemporary systems (such as OS/360 and GCOS) have been penetrated. Unfortunately, this lack of an underlying design methodology cannot be effectively overcome by ad hoc "fixes" and "security features".

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". Such a test approach is primarily limited to penetration attacks in areas indicated by the particular background and experience of the individuals involved. A security evaluation through such attempts may reveal weaknesses of a system but provide 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 prove or certify that an actual penetration attempt will overlook it at a later date. 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 outlined above 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 most system changes will not affect the kernel, so routine system maintenance will not require repeated recertification.

3.3 Practical Mechanisms

The abstract security model, to be developed by the contractor, is needed in order to evaluate the adequacy of protection mechanisms. Identified below are two techniques which can be applied in the design methodology developed by the contractor:

a. A system design is represented in various levels of abstraction. The design process transforms an initial abstract model of all the system's protection relationships (derived directly from the system's specific definitions of security, thus leading to a model that is secure by hypothesis) into subsidiary levels of abstraction. As the design progresses from level to level, the representations of the model become more specific and culminate in specific hardware features. The inter-level transformations, chosen for reasons of efficiency as well as utility, can ultimately be implemented as part of the primitive operations of the kernel, and since they preserve the initial protection relationships, we can prove that the resulting design is secure.

b. The kernel design is simplified by including only those relevant operations that modify protection data bases, but not those that merely read information that is not itself being protected against disclosure. Consider as an example the Multics demand paging system -- at some level of abstraction page table entries represent capabilities that must be carefully controlled, so the kernel will have a primitive for changing page table entries; however, the page replacement selection algorithm should not be in the security kernel.

Using this model, descriptor-based addressing available in the Multics processor hardware is seen to

offer a promising basis for a security kernel design. In terms of the first design principle (complete mediation), this addressing hardware validates each reference by a user's process to primary memory by interpreting the access specified in the applicable descriptor. The security kernel insures security through its primitive operations which are invoked by the remainder of the operating system to maintain the descriptors. Because access control is vested in the well-defined and bounded descriptor mechanism, kernel software functions should be few enough and simple enough to make certification tractable, as required by the third design principle, simplicity.

Descriptor-based isolation mechanisms (such as hardware implemented rings for Multics) can provide effective as well as efficient protection of the security kernel. Thus, as implied by the second design principle (isolation), an antagonist could have complete freedom with the remainder of the system without compromising the protection provided.

4.0 Contractor Tasks

4.1 Study and Analysis

The contractor shall perform a study and analysis of abstract representations that model the access controls required in a computer system. This analysis must include the security requirements for open use, multi-user, resource shared computer systems which process various levels of classified and unclassified information simultaneously from terminals in both secure and unsecure areas. This analysis shall take full advantage of previous technology development in the area of multi-level security in computer systems.

4.2 Abstract Model

Based on his study and analysis the contractor shall develop and describe in a technical report, a specific abstract model that will provide the basis for design of a security kernel that can be certified to be secure. See Attachment 1, attached hereto and made a part hereof.

4.3 Secure Design Methodology

The contractor shall develop and detail in a technical report, a methodology for applying this model to the design of practical computer system. The contractor shall specifically analyze the Multics system by using this methodology to assess the suitability of the Multics architecture as the basis for a certifiable secure system. As part of the methodology, the contractor shall specifically describe those classes of vulnerabilities which his system handles, and those classes of vulnerabilities which it cannot handle.

4.4 Recommendation for Prototype Application

The contractor shall recommend specific alternatives for the development and implementation of a prototype secure system based on the abstract model and design methodology developed under this contract. These alternatives shall specifically include a re-implementation of the Multics system with only minor, if any, hardware changes.

4.5 Review Meeting

The contractor shall conduct bi-weekly review meetings, as required by the Program Manager, at a location to be determined. The contractor shall present reports of progress and projected tasks for completion of the effort. Program coordination and technical guidance would be provided by the Program Manager (ESD/MCIT), 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(MCU), Data Management Office, L G Hanscom Field, Bedford, Massachusetts 01730, at the time of distribution.

ATTACHMENT I

Levels of Abstraction in a Secure Operating System Kernel

Level 0: Conceptual models, mostly patterned after the work of Butler Lampson, as interpreted by Graham and Denning, and finally as further developed by LaPadula and Bell, Popek, and Downey. These models define a schema for access control.

Level 1: A particular instance of the Level 0 Conceptual Model will constitute this level and will provide a definition of what it means for a system to be "secure". The definitions for this instance will be taken from AFR 205-1. Further and more computer related examples can be found in the appendix to the Ware Report and C. Weissman article on ADEPT - 50. This level will define the functions necessary to implement the model.

Level 2: At this level, a Multics like segmented architecture is used to provide a structure for the implementation of Level 1. The domains and ranges of the functions defined above are specified in terms of the segments, directories, access control lists, ring structure, etc. of the Multics system. Organick provides reference material on this level.

Level 3: This level describes the use of the segmented virtual memory to implement the functions of Level 2 such as the separate descriptor segment per process. Documentation for this level is available in Organick and in Bensoussan, Clingen, and Daley.

Level 4: The basic system structure used to implement paging, I/O operations and processor multiplexing, etc.

Level 5: The hardware of the HIS 6180.

NOTES: Each level is composed of functions, some of which

are implemented at lower levels. As an example, Level 2 might contain some function E which evaluates the access of some request. E might be composed of two other functions, Ψ and $\emptyset$.

$$E = \Psi \circ \emptyset$$

Where the range of Ψ is over some data base in Level 2, and the range of $\emptyset$ is over a data base in the next higher level, Level 3.