MAC 7506

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Project MAC

Computer Systems Research Division

Proposal for Continuation of Research on Engineering of a Computer System for which Security can be Certified by Auditing

Submitted to

Honeywell Information Systems Inc.

United States Air Force Electronic Systems Division
and
Defense Advanced Research Projects Agency

May, 1975

[illegible]
Jerome H. Saltzer
Principal Investigator

Joel Moses for Michael L. Dertouzos
Michael L. Dertouzos
Director, Project MAC

George H. Dummer
Administrative Director
Division of Sponsored Research

SUMMARY OF PROPOSAL

The Computer Systems Research Division of M.I.T. Project MAC proposes the continuation of its research program to make possible the certification, by auditing, of the centrally protected core of a general-purpose, remote-accessed, multi-user computer system. The original proposal, dated October, 1973, was for the 3 1/2 year period from January, 1974 to June, 1977. The initial contract between HISI and M.I.T. (actually a directed subcontract of a cost-sharing contract between AF/ESD and HISI) provided funding for the first 1 1/2 years, from January, 1974 to June, 1975. Progress during the initial 15 months has been largely as expected, so it is proposed here that the originally planned work be continued for the remaining two years. A specific budget for July, 1975 through June, 1976 is provided, together with a rough estimate of budgets for ensuing periods.

On The Need For This Work

The work carried on under this proposal is part of a program to provide security within computer systems. If successful, it would allow processing of information from different compartments and with different sensitivity levels, using logical partitions enforced by the computer system rather than physical partitions imposed from outside. Since there are substantial differences of opinion about both the need for and possibility of achieving internally-enforced security, this section provides justification for the work.

The essence of the line of reasoning is that two of the most difficult problems hampering more effective computer use today--the "software problem" and the new strategies required to harness modern hardware technology--require for their solution the automation of communication between independent computations. Yet this automation of communication, which has only rarely been attempted in traditional modes of computer use, carries also a requirement of internally-enforced security.

In 1959, the first computers that attempted multiprogramming were introduced. The purpose of multiprogramming was to obtain a higher percentage of use of an expensive component--the computer processor--by having several independent jobs ready to run at all times, and switching the processor from one job to another whenever the first job encountered a delay. Because the jobs being processed were independent, an unanticipated requirement surfaced almost immediately: the computer system had to provide isolation between the jobs, so that an accident in one did not disrupt another. Some simple form of memory protection has been part of the architecture of most computer systems ever since.

However, the memory protection was typically designed only to minimize the chance of accidents, not to guarantee perfect isolation. As a result, through the 1960's and even today certain applications have never been able to take economic advantage of multiprogramming. Processing of important records (such as payroll) or of classified information is performed by temporarily or permanently dedicating a complete computer system to the sensitive application.

With the rapid drop of hardware manufacturing costs experienced in the 1970's (and projected for the 1980's) the economic advantages of multiprogramming a processor are vanishing, and the dedication strategy--especially permanent dedication--is becoming economically reasonable. However, in the interim, computer users have discovered new reasons for independent computations to share facilities, *and these new reasons for sharing can not disappear with improvements in technology*.

These new reasons center around one idea: automation of communication between independent computations. Traditional modes of computer use envisioned little communication between users. A user presented a job to the computer and took the results away with him. If he needed a program written by a friend or a data file on a tape belonging to someone else he negotiated their acquisition outside the computer; the computer system was uninvolved in the communication. Since communication among users was carried on outside the computer, all questions of authority, classification, security, etc., were controllable by traditional, non-automated systems.

Today, probably the most significant difficulty in harnessing computers is the "programming problem". Several new techniques have been devised and tested during recent years, and are beginning to show promise of controlling the programming problem. Some of these techniques are:

1) Structured programming with higher-level languages.
2) Team programming.
3) On-line programming development systems.

Of these, the last two are, in essence, recognition that in a large-scale programming project the most crucial aspect is communication among the specifiers, designers, and implementors of the programs. This communication must begin with agreement on specifications and it continues through detailed programming when communication of problems and solutions is essential, and where central control of status and progress is needed to permit discovery of problems. The communication may consist of, for example,

1) Formal specifications of programs.
2) Exchange of completed programs for testing.
3) Exchange of test data.
4) Messages involving status of projects.
5) Integration of collections of programs.

Team programming and on-line programming development systems attack the communication aspects of the programming problem by using the computer to help automate the communication. On-line programming development systems such as TENEX (BBN), the Multics Software Factory (Honeywell), the Dynamic Modeling System (M.I.T.), the Programmer's Workbench (Bell Labs), and the Development Sub-System (IBM) are all examples of automated interprogrammer communication subsystems; all have achieved significant improvements of programmer productivity. The prototype National Software Works (ARPA) is another example of a system with similar goals.

These observations are relevant to security in the following way: since the strategies all involve automation of communication, they imply that the programmers involved must be sharing a single system, or using a network of interconnected systems; they further imply that at least minimal protection mechanisms must be provided to avoid unwanted or accidental interactions among cooperating programmers. The conclusion is that isolation mechansims with controlled communication paths are an essential part of the computer systems needed to help solve the programming problem.

A second, independent line of reasoning arises by examining the implications of evolving computer hardware technology. Two developments are emerging which will undoubtedly have a significant effect on future computer system organization and use. One of these is the complete computer on one or a few silicon chips; the other is the very large (terabit) data storage device.

Again appealing to earlier experience, the introduction of the disk file in the early 1960's brought with it a discovery that in order to produce an economical device its size had to be larger than needed for most single applications; mechanisms to protect the data of one application from harm by another application sharing the same disk file were rapidly invented. Today's situation with respect to large storage devices is slightly different: the applications which have brought such devices into existence by their very nature involve large scale data base management, with dozens or perhaps hundreds of individuals updating and retrieving information. Thus the data base management system itself is the modern communication medium which requires automated control to insure security.

The effect of the computer-on-a-chip technology is likely to be that it will soon be economical to dedicate a computer to almost every application. However, to help control the software problem, and to permit access to the large-scale data bases implied by the other advancing technology, it seems inevitable that these dedicated computers will frequently be interconnected in networks, again introducing automated communication among independent computations and the need for automating the control of that communication.

In summary, then, it is argued that new modes of computer use, based on advancing technologies, intrinsically involve automated communication among independent users, and thus that it is important to develop automated (that is, computer-enforced) security barriers to insure that only desired communication does occur.

How this project fits in

Providing automated computer-enforced security barriers by itself is a relatively straightforward engineering problem, involving first the development of a model of the security system wanted, and then the design of mechanisms to enforce that model. Weissman (1), in 1969, described such a model and design for computer enforcement of the Department of Defense military classification system. Since then, others have suggested alternate models, alternate mechanisms and hardware and software architectures to support those mechanisms.

The primary unsolved problem of computer-enforced security is to establish that the design and implementation of the enforcement mechanisms exactly match the desired model. The mechanisms involved tend to be intimately entangled with the very complex resource-multiplexing and storage management facilities of the operating system supervisor. In order to have confidence in the correctness of the security implementation, one must have confidence in a very large and complex collection of programs.

In response to this problem, the Air Force, in cooperation with ARPA and HISI, has undertaken an advanced development project that is intended to provide a working example of a certifiably correct security system implementation. The MITRE Corporation is carrying out the modeling aspect of the project, while M.I.T. Project MAC has been carrying out (and here proposes to continue) the system design aspects. Integration of these two aspects into a deliverable computer system is to be carried out by HISI.

One way to approach the problem of establishing correctness would be to start with an existing computer system which has the intended functional properties, and apply all available modern technology of system specification and program verification. One would write formal specifications for the interface of the system, and for each of its modules, with the provision that where necessary some interfaces of the actual system may need to be adjusted to make formal specification practical. The highest level interface specification would be compared with the desired model and a verification of equivalence would be sought. Then, one would attempt to verify that each program module met its own specifications, this time with the understanding that it might be necessary to completely rewrite the module in order to express its operation in a form and language susceptible to available verification techniques. The result of this sequence of steps, in principle, would be a certifiable system.

If one applied this approach directly to any existing general-purpose operating system it would almost certainly fail. One would end up with a collection of system interface and module specifications so large that verification is hopeless; in addition many systems are organized into modules so large and individually complex that verification that a module meets its specification would be impractical.

There are at least two sources of the size and complexity:

1) _Unnecessary function_. Included in the programs which perform essential supervisor functions are functions which could be done as well without the special privileges and powers of the supervisor.

2) _Incomprehensible organization_. Most supervisor programs have been designed with primary attention to efficiency and function; less attention has been paid to simplicity of organization, and almost none to organization which might be susceptible to formal specification and verification.

These two observations suggest that to develop a certifiable system one must, in addition to applying formal specification techniques, also do a substantial amount of system analysis and engineering, to develop a design that achieves the desired function more simply.

Thus the advanced development project is organized toward the goal of developing a certifiably correct version of the essential kernel of supervisor programs needed to underlie the Multics operating system. Mitre Corporation is developing examples of the formal specifications of the kernel, while M.I.T. Project MAC is performing the detailed engineering and analysis required to simplify the design of the existing Multics supervisor. The Project MAC work includes experimental or prototype implementations, to establish the completeness of the engineering. By starting with an existing system, the evolved result can be expected to be an immediately useful product.

This approach provides two possible fall-back positions, in the case that the final result, even after simplification, proves to be too large and complex for systematic verification:

1) If available verification technology is still unable to cope with the size and complexity of the resulting modules, the simpler underlying system and the existence of the formal specifications may make manual auditing feasible. Though not foolproof, a manual audit would provide at least some confidence in correctness. In addition, the modeling effort will provide a precise definition of what "correct" means.

2) If the attempt to develop complete formal specifications runs into trouble the prototype implementations of a simpler system are certain to be more reliable and auditable than the present, very complex organization. Also, any partial specifications will aid manual auditing, and provide a checklist of desired functions.

In summary, then, the goal is to achieve a system whose security features are certifiably correct, by reengineering an existing operating system that has approximately the correct organization and desired function, and at the same time developing formal specifications of the reengineered system. The project is designed so that if either verification or specification technology prove inadequate to cope with the resulting system, a partially useful result will still be obtained.

Current Status of the Project

Two earlier documents (the original proposal, and the June, 1974, Project MAC Annual Progress Report) establish the grand plan for the simplification part of the project, so that plan is not repeated here. Instead, a brief review of current status and plans for the coming two years are provided.

Five graduate students are actively working on Master's theses, all of which involve both resolution of technical research problems and checkout of the resulting ideas by experimental implementation in the Multics environment. These five are:

1) Rearrangement of the implementation of virtual processors into two layers. The first layer implements a fixed (but adequate for system use) number of virtual processors, and therefore requires no storage management. The storage management facilities (that is the virtual memory/paging programs) use several of the virtual processors, while the remainder are used by the second layer to implement any desired, variable number of virtual processors for application to customer programs. The second layer implementation is simplified by the assumption that the virtual memory system can do storage management. An interesting feature of the design is a proposed inter-process communication technique that does not require the receiver of a message to write in any shared variable; this property may simplify proofs of security in certain situations.

2) A new strategy for management of the paged, multilevel implementation of the virtual memory. This new strategy uses several parallel virtual processors, each dedicated to a narrow task and needing minimum communication with the others. The experimental

implementation uses two dedicated virtual processors, one that concentrates on choosing pages to move from primary memory to secondary memory, and a second that chooses pages to move from secondary memory to disk storage. Neither has anything to do with moving pages from disk or secondary memory into primary memory; that task is done by the customer's own virtual processor as he needs the pages. This strategy isolates the page removal selection algorithms so well that it may not be necessary to consider them part of the system that must work correctly to prevent unauthorized information release.

3) A simplified strategy for creation of processes. This project is exploring a recently-realized equivalence between the mechanics of entering a protected subsystem and the mechanics of creating a new process in response to a user's login. The goal is to make a single mechanism do both tasks, with a consequence that a large collection of special-purpose highly privileged code currently used to authenticate and log in users would become ordinary, non-privileged code controlled by the protected subsystem entry feature.

4) Rearrangement of the initialization of the operating system to simplify establishment that the starting state is correct. The question being explored here is

whether it is easier to establish that an operating system is starting from a correct state by initializing it in advance and certifying the result, which is then used as the starting point for system operation each time the system is restarted. The alternative is to use certified programs to perform the complete initialization every time the system must be restarted. The latter course (currently used in Multics) leads to the awkward need to certify programs that operate in non-standard (not completely initialized) environments.

5) Making name space management an unprotected library service rather than a protected supervisor function. The mechanics of maintaining a mapping between character-string names of segments and system segment numbers is easily confused with the mechanics of maintaining a mapping between system segment numbers and the physical segments themselves. The latter function must be protected in order to insure integrity of shared data, but the former function need not be. An experiment was undertaken to see if the difference in need for protection could be exploited; the result has been a significant reduction in the size and complexity of the protected programs of segment name management. (The reduction comes about largely from elimination of variable-length arrays of character strings from management by protected programs.)

Three further projects are actively underway, in the hands of professional staff members rather than graduate students:

6) An "infinite-buffer" strategy. This strategy would use the virtual memory as a kind of infinitely long buffer for input streams such as the ARPA network. Growing out of an earlier Ph.D. thesis, the strategy is being implemented experimentally to learn what problems might be encountered in practice.

7) Multiple virtual processors per user. Several areas of the current Multics supervisor are complicated by the current strategy of providing exactly one virtual processor per address space and exactly one address space per logged in user. Most significantly complicated is the mechanics of the "quit" feature, which is exercised when the user suddenly realizes that his computation is not proceeding as desired and should be directed to another path. A small set of experimental programs have been developed that allow testing of alternate, simpler strategies for handling "quit" and related problems.

8) Quick benchmark. In order to be sure that new implementation strategies do not have a damaging effect on system performance, a shorter, more precise benchmark test than previously available is being developed. The objective is a benchmark that tests all

aspects of system performance in less than 15 minutes, yet is repeatable (precise) to within 5% on key measurements.

The eight projects described above are in various stages of design or experimental implementation. There are several other projects that are currently under study, and which will probably be underway in the coming year. These include:

1) A project to try out a precise system specification language on portions of the operating system. The goal would be to see if some closer connection to the current work on proofs of correctness and on languages for structured programming might be developed.

2) A project to explore the tradeoffs between reliability and security. The issue here is to understand better why systematic reliability and recovery strategies seem sometimes to work at cross purposes with security and isolation strategies.

3) A project to see if management of segments as objects can be completely decoupled from catalogs or directories, making directory management an unprotected library function.

4) A project to explore the interaction between ARPANET communication protocols and block encryption strategies. Use of an ARPANET-like multiplexed channel

for all source/sink input/output streams seems an attractive way of simplifying systems, so methods of securing ARPANET-type links must be thoroughly understood.

Status of Coordination with Honeywell

An important aspect of the overall project is that to the extent feasible, new ideas and experimental implementations should be considered as inputs and prototypes for the continuing development of Multics by the Cambridge Information Systems Laboratory (CISL) of Honeywell.

Contacts have been maintained (and new ones developed) to ensure that this transfer of technology takes place. For example, the completed project of removing the dynamic linker from the protected supervisor and the almost-completed project of removing name-space management from the supervisor are being integrated by CISL into a forthcoming release of Multics. Most of the remaining current projects are similarly expected to evolve from experimental prototypes into parts of the delivered system. A CISL project to reduce the "fragility" of the storage system by rearranging physical device management is of considerable interest to us because it makes several potential simplification tasks much easier to accomplish quickly.

Budget

The overall activity plan is to maintain approximately the current pace of operation for the fifteen months ending September 30, 1976, and then begin a phase down of Project MAC activity during the period October 1, 1976 - December 31, 1977, at the end of which period activity would be around 10% of the present level. By that time, Honeywell activity in incorporating the results into a production version would be at a high level.

The total amounts currently programmed by Project MAC for these periods are as follows:

FY1976	(7/1/75-6/30/76)	$410,000
FY1976T	(7/1/76-9/30/76)	109,000*
FY1977	(10/1/76-9/30/77)	280,000
FY1977T	(10/1/77-12/31/77)	25,000
FY1978	(1/1/78-12/31/78)	50,000

These amounts assume that salary, benefits, and overhead will rise at a rate of about 8% per year. On the following page is a detailed budget for FY1976.

In preparing this budget it has been presumed that arrangements for access to the Honeywell development computer system will be provided by HISI rather than purchased from HISI with contract funds. Approximately 180 hours of development time will be needed.

* The three-month budget periods are provided on the assumption that funding of this project will be coordinated with the U. S. Government plan to shift its fiscal year to match the standard calendar year.

Similarly, it has been presumed that arrangements for access to the M.I.T. service Multics computer system will be provided by AF/ESD, rather than purchased with contract funds. An expenditure of approximately $125,000, at current M.I.T. rates, is anticipated.

Project MAC Simplification/Certification Project

Estimated Budget for 7/1/75 - 6/30/76

Salaries and Wages:

	Head Count	Full Time Equivalent	Direct Cost	Direct plus Indirect Cost*	
Faculty and Research Associates	5	2.2	$44,220	$ 81,719	
Staff Programmers	5	3.0	49,800	92,030	
Graduate Students	14	8.5	79,050	130,828	
Undergraduate Students	6	1.0	8,000	13,240	
Support Staff (secretarial)	3	1.0	8,400	15,523	
Administrative Staff			14,097	26,045	
Total salaries, wages, overhead and benefits				$359,385	$359,385

* Indirect cost includes overhead at 65.5% and Benefits (excluding students) at 19.3%

Computer Time:

Development time:	to be supplied by Honeywell at its Cambridge site
Service time:	to be supplied by the Air Force at M.I.T. Information Processing Center

Other:

15 terminals and communication equipment at $150/mo.	$ 25,200	
Travel (16 trips at $500)	8,000	
Reproduction (160,000 sides at $.025/side)	4,000	
Telephone ($180/mo. basic service and $100/mo. tolls and message units)	3,360	
Miscellaneous (alterations, maintenance and repair of equipment, office and lab supplies, books and reports, page charges, postage and shipping)	9,750	
Total Other	$ 50,310	50,310
Total estimated budget, 7/1/75 - 6/30/76		$409,695

Technical Personnel

Faculty and Research Associates

Jerome H. Saltzer, (Principal Investigator)
Associate Professor

Michael D. Schroeder, Assistant Professor

David D. Redell, Assistant Professor

Liba Svobodova, Assistant Professor

David D. Clark, Research Associate

Staff Programmers

Nancy Federman

Robert Mabee

Rajendra Kanodia

Douglas Wells

Kenneth Pogran

Graduate Students

Richard Feiertag, Instructor

Douglas Hunt

Philippe Janson

Harry Forsdick

David Reed

Warren Montgomery

Allen Luniewski

Andrew Mason

Harold Goldberg

Andrew Huber

Eugene Ciccarelli

Linda Scheffler

Steven Kent

+ 1 new graduate student