ELECTRONIC SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
HANSCOM AIR FORCE BASE, MASSACHUSETTS

MCI-75-2

September 1972

SATIN COMPUTER SECURITY

INFORMATION SYSTEMS TECHNOLOGY APPLICATIONS OFFICE
DEPUTY FOR COMMAND AND MANAGEMENT SYSTEMS

This technical report has been reviewed and is approved for publication.

Mark H. Richardson

MARK H. RICHARDSON, 1Lt, USAF
Techniques Engineering Division

Roger R. Schell

ROGER R. SCHELL, Major, USAF
Techniques Engineering Division

FOR THE COMMANDER

[illegible]

ROBERT W. O'KEEFE, Colonel, USAF
Director, Information Systems
Technology Applications Office
Deputy for Command & Management Systems

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER MCI-75-2	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) SATIN Computer Security		5. TYPE OF REPORT & PERIOD COVERED Interim Report
		6. PERFORMING ORG. REPORT NUMBER MWP 4445
7. AUTHOR(s) S.B. Lipner		8. CONTRACT OR GRANT NUMBER(s) F19628-73-C-0001
9. PERFORMING ORGANIZATION NAME AND ADDRESS The MITRE Corporation Box 208 Bedford, Mass. 01730		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Project 6190
11. CONTROLLING OFFICE NAME AND ADDRESS Deputy for Command & Management Systems (MCV) Electronic Systems Division, AFSC L.G. Hanscom AFB, Bedford, Mass. 01731		12. REPORT DATE 27 September 1972
		13. NUMBER OF PAGES 25
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Deputy for Command & Management Systems (MCI) Electronic Systems Division, AFSC L.G. Hanscom AFB, Bedford, Mass. 01731		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

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

18. SUPPLEMENTARY NOTES

Since this material is of an interim nature, this document does not qualify as an ESD Technical Report, and it is NOT AVAILABLE THRU DDC.

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

Communications Processors	AUTODIN
Computer Security	Certification
Security Kernel	

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

This paper deals with the computer security aspects of the SAC Automated Total Information Network (SATIN). Threats to communications processors are identified, and historical countermeasures reviewed. The paper examines problems that will arise if security is not an explicit consideration in the design of the SATIN communications processors. Use of segmented memory, multi-state processor hardware and a separable software "security kernel" is recommended for all SATIN communications processors. Early implementation and certification of a SATIN BCP prototype are proposed.

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

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

SECTION I

INTRODUCTION

OVERVIEW

This paper discusses the computer security issues that must be addressed in designing the SAC SATIN network. As a discussion of computer security, the paper is primarily concerned with the hardware architecture and software design and implementation for the network's communications processors. The issues of link communications security and of physical and procedural security are not addressed, except as they relate to hardware and software. The discussions in this paper are preliminary in nature and will be expanded as the network design evolves.

Section II below provides background by discussing potential threats to the security of communications processors and the ways they have been countered in AUTODIN. Section III outlines some problems and potential vulnerabilities that will be of concern in the design and development of the SATIN communications processors. Section IV identifies some approaches to providing security controls for the SATIN communications processors. The remaining subsection of Section I presents a few definitions of terms that will be used in the remainder of this paper. Despite the inclusion of these definitions this paper does not stand alone; the reader is assumed to have some familiarity with the issues of computer security.[1]

DEFINITIONS

The following definitions are taken (more or less verbatim) from Anderson.[2] They seem to have gained some degree of acceptance in discussions of computer security.

---

[1] J. P. Anderson, Computer Security Technology Planning Study, James P. Anderson & Company, Fort Washington, Pa., October, 1972, ESD-TR-73-51

[2] Anderson, J. P., AF/ACS Computer Security Controls Study, James P. Anderson and Company, Fort Washington, Pa., November 1971, ESD-TR-71- 395

A threat is a deliberate covert attempt to:

(a) obtain information from a system;

(b) inject spurious information into a system; or

(c) deny the user population the services of a system.

An attack is a specific formulation and execution of a plan to carry out a threat.

A vulnerability is a flaw in the design or implementation of a system that renders it susceptible to attack.

A successful attack may require exploitation of several vulnerabilities. For example, a general-purpose computer may have supervisor services that do not adequately check input parameters, but users may be forbidden machine-language access to the supervisor services. An attack may then involve:

(1) exploiting compiler vulnerabilities to insert machine-language code; and

(2) exploiting supervisor service vulnerabilities to gain control of the system.

SECTION II

THREATS AND COUNTERMEASURES

INTRODUCTION

This section identifies the general categories of threats to which a communications processor handling sensitive or classified information may be subject. These threats are restricted when compared to those that apply to a general-purpose (user-programmed) computer, but still bear consideration during processor design and implementation. The second subsection below provides an overview of the security control mechanisms of AUTODIN, a system whose communications processors are required to protect classified information.

THREATS TO COMMUNICATION PROCESSOR SECURITY

The following paragraphs enumerate three classes of threats to the security of information in a communications processor or network of communications processors. Many different attacks may be subsumed under each class of threat. The choice of an attack against a specific system is governed mainly by the vulnerabilities of that system. These classes serve to provide an overview of potential threats to communications processors, and to exclude some classes of threats. For example, users do not normally program communications processors and, for this reason, no threats based on user programming are included.

User Input Threats

The simplest class of threat to the security of information in a communications processor involves user input of a message that causes compromise or falsification of classified information, or denial of system service. This threat does not apply to a communications processor with "complete and correct" security control mechanisms; since the communications processor's user (subscriber) does not insert programs himself, he can only take advantage of errors of trap-doors (see next subsection) in the existing hardware and software. Indeed, the function of an agent in a communications processor programming team would typically be to "set up" user input threats, rather than to act against the system himself.

A simple example of a user input threat is the possibility of inserting a long or ill-formatted message that invokes a software error and causes the processor to crash. If a system allows retrieval of previously delivered messages (as do most store-and-forward processors), it may be possible for a subscriber to retrieve a message that he has no business seeing. Entry of a message that is too long, has too many addressees, or ends improperly may cause errors in a communications processor's buffer management routines and result in improper handling of messages adjacent in time or storage to the one in error.

User input threats were considered at length in the design of AUTODIN, and numerous countermeasures were included. The design philosophy of AUTODIN is discussed later in this section. The paragraphs below address the threats associated with the communications processor programming and design staff.

Design and Programming Threats

The most severe threats to security in a system of communications processors come from the programmers and others who implement the system. Such programmers can deliberately insert trap-doors -- sections of code intended to bypass critical security controls under known circumstances. For example, a program might detect a message having a specified addressee, precedence, and classification, and generate an "unclassified" duplicate destined for an appropriate addressee. Alternatively some messages might be altered in selected ways (falsification of data) or caused to disappear from the communications processor (denial of service).

The reader of the preceding paragraph may envision an isolated and organized section of code -- perhaps beginning with a comment line "*** DIVERT MESSAGE TO AGENT'S TERMINAL ***." In fact, efficient (fast, compact) programs are often singularly obscure of purpose and can conceal their true effect from their authors. Hiding a small selective penetration in a large assembly-language program is thus not especially difficult. Attacks aimed at communications processor components that are not apparently security-related can be even more effective. A distributed update of an obscure master mode procedure in a vendor-supplied operating system is fairly unlikely to be inspected for its true security implications. A special update to processor or control unit microprograms can bypass security controls and will likely be ignored by programmers (who see it as part of the hardware) and by hardware staff (who consider it a kind of software). The latter attacks have the advantage that they can be performed by mail without inconvenience or danger to the agent involved.

In implementing a concealed trap-door of the sort mentioned, an agent could elect to defer providing a full message rerouting capability. Instead, he might include only the handful of instructions needed to detect a "patch" message and replace code at a location specified by the messages with newly specified instructions. This alternative provides in the communications processor a limited user programming capability. The agent can replace the communications processor programs (or such of them as he wishes) remotely by message input.

To prevent design and programming threats, the communications processor environment in its entirety must be safeguarded. The communications processor program must be certified and protected as though classified. All programmers must be cleared. Even the assembler or compiler used to prepare object programs is capable of inserting a trap-door, and must be certified for its security role. This approach to safeguarding the communications processor environment is the one followed by AUTODIN.

Host Computer Threats

A new class of threat is introduced in a network of communications processors that support attached (host) data processing computers. The exact nature of this threat has not been defined as clearly as those in the previous classes and, in some sense, this subsection will "talk around" the nature of the threat. However, it does seem that a potential threat exists, and it is clear that SATIN is a potential point of attack.

The general nature of the host computer threat involves too much trust between the host and communications processors. The host processor knows that the communications processor is cleared for top secret (for example) and is willing to pass it data of that level. However, the communications processor also serves uncleared subscribers, and can compromise the top secret information without detection by the host. Similarily, the communications processor sees the host as cleared for top secret, and will accept data up to that level. If the host sends the communications processor a top secret message, marked as unclassified and destined for an uncleared user, the communications processor can do naught but deliver it. In this case, a host processor security failure propagates through the network.

The first class of network problems above is simply a manifestation of inadequate communications processor security. The second, however, reflects a host computer problem that is relatively likely. The communications processor cannot examine the content of a message to discern its true classification. However, it is possible for the communications processor to "selectively distrust" the host computer and treat all of its output messages as tentative top secret. In this case, the host computer is assumed to be unsuitable for secure multilevel operation, and human review (external to the host) is required before the tentative top secret data can be downgraded.

The basic problem reflected above is that each communications processor in a network tends to trust its neighbors. While the nodes of a network appear to exercise their security controls in series (one after another), action by any node to change message classification can defeat the remaining checks in the series. Thus, as in the present manual security system, the responsibility for the original classification of material rests with the originator -- the host computer.

AUTODIN SECURITY MEASURES[3]

System Characteristics

AUTODIN is a message switching network that provides for interchange of narrative and data messages among a large community of DOD subscribers. Some AUTODIN subscribers are uncleared, while cleared subscribers may exchange messages classified up to top secret. The network includes a number of communications processors connected to each other by medium-speed lines and to subscribers by low-and medium-speed lines. The main subscriber methods of "controlling" the communications processor software involve entering new messages to be routed and delivered, and requesting retransmission of messages already delivered. In each case, subscriber-entered parameters such as addressee, classification, precedence, and date-time group direct the operation of the message-switching software.

The AUTODIN communications processors operate in a true multilevel security mode, handling classified information and supporting cleared and uncleared subscribers. There is no attempt to partition classified buffers in core or drum stroage from unclassified ones; all are drawn from a common pool on an as-needed basis. Similarly, a single software package processes all messages, classified and unclassified. The security checks performed by this package provide AUTODIN's major security control mechanism.

---

[3]Much of this material is taken from an outline, Computer Security Aspects of AUTODIN, prepared by Miss H. C. Faust of NSA and presented to the ESD Panel on Computer Security Technology on 18 May 1972.

Security Controls

The overall philosophy of AUTODIN's security controls has been to include multiple redundant checks throughout message processing. Typically the checks are performed by separate instructions interpreting distinct tables in the AUTODIN programs, and apply to different parts of the message. For example, the classification of an incoming message is indicated both by the fourth character of its header and by four "classification redundancy" characters later in the header. One routine checks the validity of the initial classification character, while a second routine compares all five classification characters. Should the validity check fail or the characters not be identical, the message is rejected. A similar set of redundant checks is applied to the (80-character) message buffers that move through the AUTODIN communications processor. Each buffer carries with it a classification character, and the classification of every buffer of a message must be the same at every point of the message's travels among core, drum, and communications circuits. This sequence of checks insures that an error in handling a linked list does not cause undetected intermixing of buffers from two messages of different classifications.

A second AUTODIN security technique involves restricting user-input control parameters to members of a predefined set. For example, the classification characters referred to above may be A, T, S, C, E, X, or U but no others. User choices of routing (destination) and precedence are restricted in a similar way. The reaction of the software to parameters outside the defined set is specified and tested. Explicit controls also apply to message length and the characters that frame various parts of the message. The software reaction to unexpected parameters is required to be initiation of an error routine, rather than an attempt to "carry on" and make sense of the parameters in some creative and potentially dangerous way.

Subscriber inputs to AUTODIN are carefully segregated from the AUTODIN programs. A subscriber might input a new message processor program as the text of a message, but would have no way to force the processor to transfer to his program. Programs, tables, and message buffers are separated from each other in processor memory. Thus insertion of message text in a program area would require a read from the wrong drum or into the wrong memory area. For the text read to contain instructions would require an extremely unlikely accident or the existence of a programmer trap-door.

Message retrieval facilities in AUTODIN appear, at first glance, to provide a way of obtaining information without authorization. However, all retrieval requests are handled manually and a message, once retrieved, receives the same processing as it would on initial entry to AUTODIN. Thus an unauthorized subscriber might initiate retrieval of a message, but the retrieved message would be delivered only to its initial (authorized) addressee.

A final factor leading to the high degree of security achieved by AUTODIN is the network's high degree of uniformity and stability. While there are large numbers of terminals attached to AUTODIN, all comply with a tightly specified set of interface standards and message formats. Thus there is no need for large computer programs that select among numerous alternative message formats and line disciplines, then perform required processing for each alternative case. The basic message processing sequence and security checks apply uniformly to all messages and terminals. While new terminals are frequently added to AUTODIN (and old ones removed), a terminal change requires only a new table entry in the AUTODIN software. No program changes are required and the table entry is highly formatted and well-specified. The AUTODIN software therefore tends to be straightforward and stable. There are no elaborate alternative paths or frequent software changes that could serve to conceal a (deliberate or inadvertent) circumvention of the AUTODIN software security controls.

The AUTODIN security measures outlined above combine to achieve two effects:

(1) It is extremely unlikely that a subscriber will cause a security compromise from his terminal.

(2) A software trap-door intended to bypass AUTODIN's security mechanisms in a useful way would have to impinge on message parameters processed by a number of separate program modules and would thus be relatively large and detectable.

The software trap-door alluded to above is made more difficult to insert by AUTODIN program and table preparation and checkout procedures. These procedures provide for high-level clearance of all programmers, two-man review of all communications program updates, and extensive software checkout.[4] Thus it is unlikely that a single programmer agent-in-place could successfully insert a trap-door in the AUTODIN software.

---

[4] This list of precautions makes an interesting contrast with the alternative of using an off-the-shelf software system with little or no review or examination of individual programs.

SECTION III

SATIN COMPUTER SECURITY PROBLEMS

INTRODUCTION

This section describes some of the technical problems of computer security and points out their impact on the planned SATIN communications processors. The first subsection below is devoted to a discussion of technical problems in any computer that must provide security. The second subsection outlines the applicability of major problem areas to each class (BCP, HCP, RCP) of SATIN communications processors. While it is premature (in the absence of detailed system designs) to perform a vulnerability analysis of the SATIN communications processors, the discussions below can help system designers to avoid requiring major vulnerabilities in the processor hardware and software.

COMPUTER SECURITY PROBLEMS

The paragraphs below describe some technical problems in computer security that must be addressed by the SATIN communications processor design. The most significant and pervasive problems are the first two: the difficulty of certification and the impact of system complexity. The remaining problems are listed because they tend to be major contributors to the difficulty of certification and the growth of complexity.

Certification and Testing

If an organization is to operate a computer system with multiple levels or types of classified data and with a community of users having diverse clearances, it must undertake to certify that the system restricts each user to that data which he has need to know. Evolving regulations and directives provide an organizational framework for certification, but little technical guidance that would help determine a system's adequacy. Some certification attempts now underway are based on the concept of a test team or penetration team. Such a team is given access to the system and its documentation and directed to "find the holes." Thanks to the underlying weakness of most current systems, test teams have been quite successful in finding "holes." In such cases, certification is usually denied and the system continues to run in a closed (not multilevel) environment.

The danger and difficulty of the penetration team approach comes from the question, "how much is enough?". If a test team finds one or five or a hundred ways of penetrating a computer system, one can say with certainty that the system is not secure. But if the team finds no holes, or if the system builders repair the ones discovered, that does not imply that the system is secure. The strongest statement that can then be made is that the holes (if any) in the system are well hidden. The operator of such a system is betting that either:

(1) there are no holes in his system; or

(2) a potential attacker is less clever and persistent than the test team.

With current complex systems (see below) the first possibility is an unlikely one.

If a system is certified "hole-free" by an ad hoc test approach, its operator must ask about the impact of changes and updates. If one has certified a computer system (hardware and software) as an integrated whole, any change requires recertification of the entire system (as in AUTODIN). While a very simple change may have a restricted and obvious effect changes of larger scale rapidly develop the ability to obscure their effects -- both functional and security. Thus, at some ill-defined point, one must start certification from the beginning after a change has been implemented.

The paragraphs above point out the difficulties of certifying the security of any system, and the special problems of recertifying a highly integrated system. Clearly, one would like an orderly approach to certifying a system so that it could be confidently asserted that the certification was complete. An approach to isolating security controls from functional parts of the system would help by allowing functional changes and evolution to proceed without requiring recertification. If security controls could be minimized as well as isolated, certification or testing would also be simplified since the security controls would tend to become obvious and understandable. However, the isolation must be absolutely complete and effective, lest one leave in the security controls a trap-door suitable for entry from any point in a large uncertified portion of the system.

Complexity

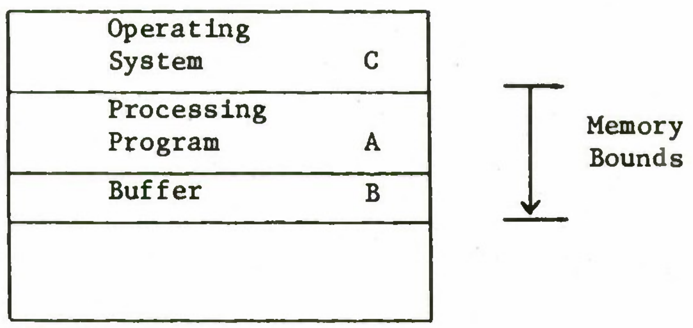
The paragraphs above mentioned the influence of complexity in the development of uncertifiable computer systems. In this context, complexity refers to the tendency of large interrelated blocks of software to become inseparable from a system's security controls. Thus the would-be certifier must understand every state and function of programs that (he hopes) have nothing to do with security. The danger is that one such program will in fact have a (negative) security function.

Complexity in application programs appears to be a natural result of functional diversity: to do a complex job, one writes a large complex program. However, the impact of complexity on security results from the inability of hardware/software systems to separate security from other system functions. The following paragraphs address the problems of current hardware and software.

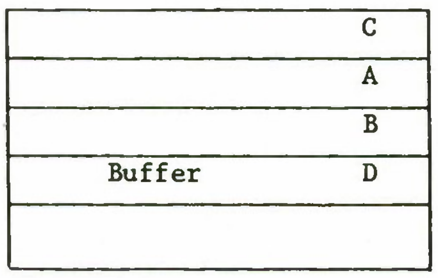
Off-The-Shelf Hardware

Complexity in modern computer systems results mainly from the hardware on which those systems are built: while one might build inadequate software on better hardware, the bulk of existing hardware simply fails to support anything but complexity. Typical third-generation computer systems provide some form of memory protection (write and read) and relocation, plus a set of two processor states (privileged and slave). In the more privileged processor state, a process (program in execution) can issue input/output instructions, reset the memory protection, and control the entire state of the processor. In slave state, a process can only execute ordinary arithmetic and logical instructions on data within its memory partition. Some low-cost minicomputers do not even provide the level of protection described above.

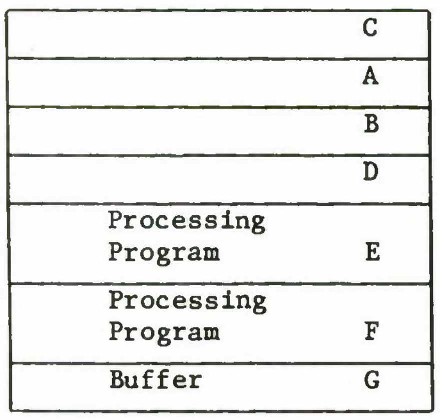
If a computer operating system is required to provide security controls in an environment such as that described, there is good reason to believe that complexity must result. As an example of this evidence, one may consider memory, access controls, and processes in a simple communications processor. Figure 1a depicts the memory of such a processor, including a processing program, message buffer, and operating system. The processing program is operating on the buffer, and the memory protection boundaries are set as shown. If executive services are needed by A, it appeals to the operating system which acts on B using its ability to address all of memory. Now assume (Figure 1b) that program A has finished with buffer B and wishes to act on buffer D. There is no way to set the memory base and bounds register so that A can access itself and D but not B. A might appeal to the operating system to exchange B and D, but this is exactly the sort of privileged operating system complexity (and functional growth) that the example is intended to demonstrate. Additional of a second or third application program (Figure 1c) makes more apparent the potential complexity of the operating system as



1a. Configuration Processing Buffer B



1b. Configuration - Ready to Process Buffer D



1c. Configuration - Additional Processing Programs

Figure 1. Memory Configurations to Show Problems of Conventional Memory Protection

possibilities arise for each program to be granted access to some buffers and denied others. A requirement for limits on interprogram communication (calls, returns) even further complicates the problem.

The provision of only two processor states in conventional computers makes more difficult the isolation and minimization of security controls. While operating system designers might put operating system functions requiring protection but not privilege in separate bounds-protected programs, current hardware does not provide convenient communication between a user program and such a separate operating system routine. Such communication must be mediated (at some cost) by the privileged portion of the operating system. The cost and difficulty of this approach are so great that most operating system functions wind up in the privileged portion of the executive.

Finally, the restriction on input-output instructions requires that an elaborate system of input-output control programs be included in the privileged portion of the operating system. These programs are complex, and have the upsetting habit of requiring modifications whenever a new input-output device is to be added to the computer.

The preceding paragraphs have illustrated the way in which inadequate hardware leads to overly complex operating system software. Practical illustrations of this effect are provided by most current commercially available operating systems.

Off-The-Shelf Software

The subsection above discussed the complexity that appears in an off-the-shelf computer's operating system as a result of the computer's architecture. To this complexity is added a basic lack of consideration for security in operating system design. The result of this lack is often increased complexity coupled with dispersion of security controls throughout the operating system.

A typical modern operating system is based on a set of interconnected tables or list structures. These list structures describe the state of tasks in execution, main memory, input-output devices, and files on secondary storage. Any change in the status of a process is reflected in one or more tables. The security problems associated with such a structure are twofold:

(1) Almost every table and table entry has some implicit or explicit security role.

(2) Tables are accessed "as needed" from all parts of the operating system.

In addition to these two underlying problems, modern operating systems typically have no consideration of such notions as classification, clearance, or need to know. The two underlying problems result in dispersion of security control (or lack of control) throughout a system, while the lack of explicit security control results in potential security violations that are legal system operations. (For example, read a top secret file and write its contents to an unclassified file.)

Modifying an off-the-shelf operating system for security is a grim and ineffective business. At best it requires an examination and rewrite of all security related code -- usually the entire operating system. Such efforts have been marked more by their cost than their security. At worst a modification may put a thin veneer of "security features" over a deep, complex, and unsecure structure. Such features may be suitable for labeling output, but typically provide no protection.

A final software aspect of computer security concerns languages for system implementation. Most operating systems and communications programs have historically been written in assembly language. In addition to its known disadvantages for training, documentation and maintenance, assembly language provides a ready vehicle for obscuring the security implications of a program. The alternative, higher-level language for system programming, is often claimed to be costly of space and time. However, most recent experience and evolving commercial practice tend to contradict these claims.

SATIN PROCESSOR SECURITY PROBLEMS

This subsection identifies those security problems discussed above that appear likely to apply to each class of SATIN communications processor. Security implications of HCP, BCP and RCPs are discussed in turn.

Headquarters SAC Communications Processor (HCP)

The Hq SAC HCP is planned to be a medium scale communications processor providing interfaces among numerous computers, communications networks, and terminals. Its functions include message switching, front-end processing and terminal control. The net effect of this combination of functions and interfaces is to require a large amount of application-level complexity. If this complexity is allowed to impact (or be integrated into) the security-related portion of the HCP operating system, the operating system will become very large and any meaningful certification impossible. Further, the computer communications and terminal environment at Hq SAC is a relatively dynamic one. If this dynamism reflects to the security portion of the HCP software, the recertification process will be almost continuous.

As was mentioned in Section II, the primary threat to the SATIN communications processors is from their programmers and implementors. In practice the danger to the HCP is that, if its security controls are not isolated, minimized, and effective, the class of programmers and implementors will become very large. Not only will it encompass those individuals who code "security features," but also all others inside and outside of SAC who program the operating system, applications programs and even compilers. In effect there is no clear way to prevent any individual who programs any part of the system from attacking (probably with good success) its security.

Base Communications Processors (BCP)

The BCPs (and NAF HCPs) provide similar types of interfaces and functions to those of the SAC HCP. The diversity of BCP functions and the number of interfaces are somewhat lower than those for the HCP. However the BCP, unlike the HCP, is to be a minicomputer. Thus there is a danger that its hardware will provide completely inadequate support for security control. In this circumstance, there is no choice but to certify all BCP software (not clearly a possible task). The dangers mentioned under the discussion of the HCP apply even more strongly in this case.

Regional Communications Processors (RCP)

The primary functions of the RCPs are packet switching and adaptive routing. This function in itself is a straightforward one and security of a "pure" packet switching RCP could probably be assured by techniques like those used by AUTODIN (including clearance of all programmers and certification and control of the entire software development environment). However, the RCPs, in addition to packet switching, perform AUTOVON automatic dialing, interface to the AABNCP, and may interface with other command control networks. Thus the RCP begins to have multiple functions and interfaces similar to those of BCPs and HCPs. In these circumstances, the same security problems described above apply to the RCP.

SECTION IV

SATIN COMPUTER SECURITY APPROACHES

INTRODUCTION

This section describes technical approaches to providing certifiable security controls for the SATIN communications processors. The first subsection below presents a brief discussion of alternative approaches to providing SATIN computer security. The second subsection identifies specific techniques applicable to the SATIN communications processors.

The last subsection of Section III pointed out the essential similarity of the security problems presented by RCPs, BCPs and HCPs. All perform diverse functions and support numerous interfaces. Thus, all will respond to similar security techniques. The techniques identified below, implemented on processors of appropriate size and power, should be equally applicable to RCPs, BCPs, and HCPs.

ALTERNATIVES

The subsection below identifies techniques for providing protection of multiple types of classified data in a single communications processor that supports a diverse collection of users. Alternatives are available to the application of these techniques. These alternatives involve:

(1) Clearing all terminals, terminal communications, and users for all data processed in SATIN.

(2) Dividing the network so that a separate subnetwork is provided for each classification, perhaps allowing one-way communication from subnetworks of lower classification to those of higher classification.

Both of the alternatives outlined are highly secure, relying basically on personnel, physical and communications security rather than computer programs. Ample precedents are available for certification of both approaches, and no new techniques are required.

Unfortunately both alternatives identified above have several disadvantages. The first (total clearance) requires numerous communications security devices, cleared areas, and personnel clearances. The cost of this approach can be quite high. Further, some areas that require access to SATIN (as currently planned) are not amenable to clearance; in effect, this alternative creates a secure network that spreads without limit.

The second alternative identified (divided network) limits the flexibility of application of SATIN. Some areas will require multiple terminals for read/write network access at several levels of classification. Computer interfaces will probably have to be implemented with manual review and intervention. The number of communications processors required by this alternative will be greater than that for an integrated network. This alternative seems costly and fails to meet major SATIN objectives.

SATIN SECURITY TECHNIQUES

The paragraphs below identify techniques that can be applied to the provision of security in an integrated network that meets SATIN requirements and objectives. These techniques are not detailed completely below, but are promising and have been investigated for application to general computer security problems. Each technique identified provides a part of the solution to the SATIN computer security problem. Together they offer the possibility of a certifiably secure network.

Choose Appropriate Hardware

Section III identified security problems that result from the use of inadequate processor hardware. Alternatives to such hardware are available "off-the-shelf." These alternatives are characterized by two kinds of features:

(1) Provision of a way of isolating security control software from other executive and application software components.

(2) Provision of a way of flexibly limiting the context (memory addressing domain) of executive and application programs.

The practical implementation of these features requires provision of multiple processor execution states (at least three) and some form of memory segmentation. The following paragraphs describe the effect of providing these features.

Memory segmentation allows an executive program to establish the address space perceived by a process. Unlike ordinary base and limit registers described in Section II, segmentation allows efficient and dynamic control over process context without requiring unnatural and complex movement of information. Figure 2 depicts a memory configuration under segmentation. In Figure 2a, a program A accesses data in a buffer B. The program has read and execute access to itself and read and write access to the buffer. Both A and B may occupy any physical address in memory. In Figure 2b, a second program, C, has read and execute access to itself and read-only access to B. In Figure 2c, program A has read-write access to a new buffer D. The key capability provided by segmentation is that of providing a process with controlled (read, write, or execute) access to any information in memory. Hardware aids to segmentation allow very rapid switching from one set of accesses to another, and provide for checking of the validity of every reference to memory at almost no cost.

Multiple execution states combine with segmentation to provide a foundation for a secure system. With such multiple states, control over security (corresponding to control over what segments can be accessed at any time) can be isolated into a small security control package (a kernel). The bulk of operating system services are provided by a "supervisor" that has considerable power over application programs but no security role. Application (or functional) programs are relegated to a third unprivileged execution state. Input-output operations, normally restricted to the privileged part of the operating system, may be performed by supervisor or functional programs, depending on the exact nature of the processor hardware and on operating system design.

The costs of processors with the features described need not be great. In a minicomputer (BCP), the cost added need be no more than a few thousand dollars per CPU over more conventional equipment.

Design Software for Security

The sections above have provided ample indication of the importance of early consideration of security in designing a computer system. The discussion in the preceding subsection described a hardware foundation for security controls. The software for the SATIN communications processors must apply the hardware foundation to provide workable and certifiable security controls. Such controls must not only protect classified messages from unauthorized processes, but must also be sufficiently isolated from processor configuration and function to prevent a requirement for constant recertification.

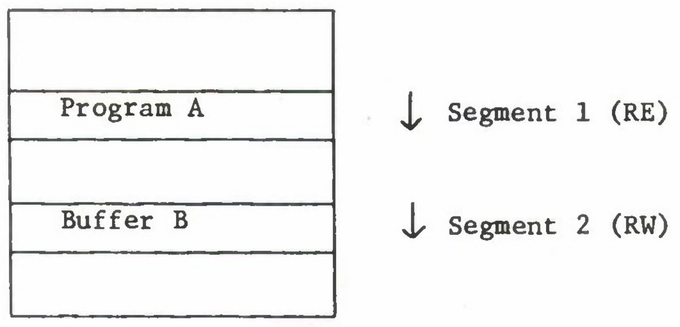


Figure 2a. First Program and Buffer

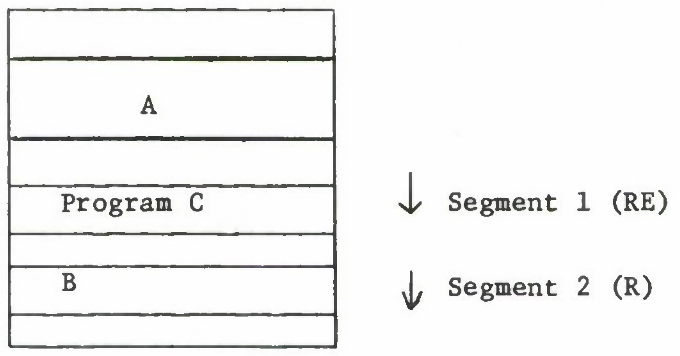


Figure 2b. Second Program and Buffer

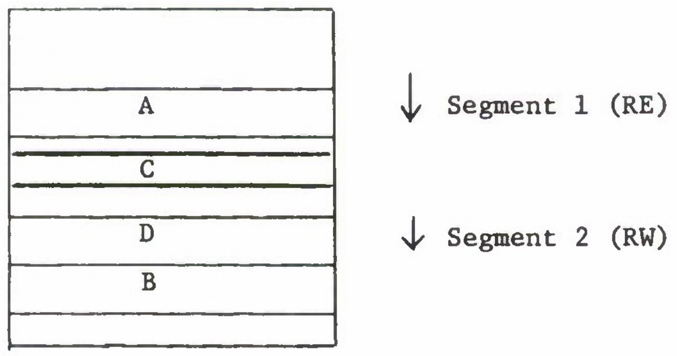


Figure 2c. First Program and New Buffer

Figure 2. Access Controls with Segmentation

Figure 3 depicts an overall software structure capable of providing the required security controls. As indicated in the discussion of hardware for security, the software structure is divided into three components or layers. The lowest layer is a security control "kernel" whose sole function is security management. The second, or "supervisor" layer performs resource allocation and "recommends" actions to the kernel. The supervisor cannot implement resource allocation decisions that have security impacts without having those decisions ratified by the kernel. The third, or functional, layer performs the actions required by the communications processor users -- message switching, front-end processing, and terminal control. In general, the functional programs operate on data (buffers) of one classification at a time and appeal to the supervisor and kernel for interprocess communication and privileged operations.

To provide stable, certifiably secure SATIN communications processors, the kernel discussed above should be designed early, and its security evaluated (see next subsection). The kernel must be sufficiently flexible to accommodate SATIN functions and configurations with a minimum of change and reprogramming. In general, common kernel programs will learn about their specific environment through formatted tables. The programs of the kernel must be uniform, although each processor will probably require a new set of table entries.

Plan and Undertake Certification

If SATIN is to perform its security functions with initial operation, the approach that will lead to system certification must be identified early and followed in parallel with design and implementation. If a security kernel is to be used, it should be designed early, and the strongest analytic tools available applied to evaluating the security of the design. A prototype kernel should be implemented and tested, both analytically and by attacks. To the extent that the kernel is successfully minimized, such testing and evaluation should be simplified.

Cognizant agencies should be made aware of security plans as the kernel prototype is designed and implemented. The certification plan for SATIN should be made explicit, and reviews applied to a design, not solely to a finished system. Test plans should be developed, emphasizing examination of specific (provably) required security controls, rather than an endless search for "holes." In this way, a closed certification process can be followed, with the potential for developing a secure system in a known time.

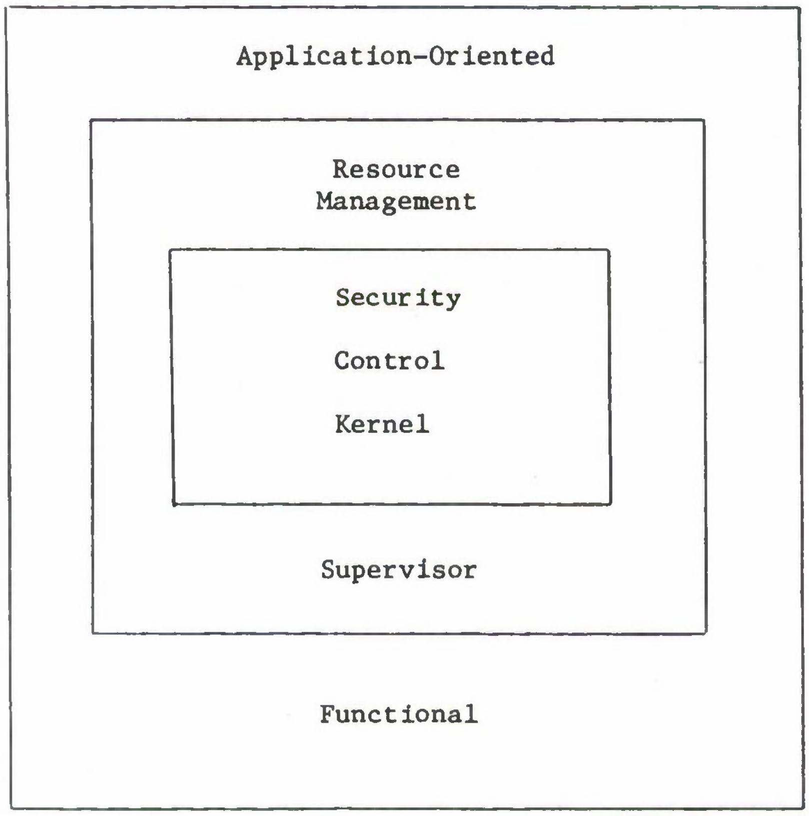


Figure 3. Secure SATIN Software Structure

Use High-Level Language

At several points above, this paper has emphasized the utility of assembly language code for concealing security flaws. While a security kernel should be small and offer few places for concealment, the use of high-level languages throughout the SATIN communications processors still seems appropriate. In addition to advantages of program visibility, such usage should offer improved documentation, training and maintenance during the life of SATIN.

While the use of a high-level language is desirable, it is not mandatory for security. Thus this subsection is a suggestion, while those preceding it were strong recommendations.

SECTION V

SUMMARY

The SATIN communications processors, while not as vulnerable as general purpose user-programmed computer systems, present several computer security problems. Current common computer system design techniques do not appear especially promising as ways of solving the SATIN problems. For this reason, use of hardware possessing multiple processor states and memory segmentation is recommended. A security kernel can be built on this hardware that will centralize security functions in a configuration-independent module. Certification of this single kernel will then suffice for secure operation of all SATIN processors using the kernel (BCPs, RCPs, or HCPs). Early design and implementation of such a kernel, leading to certification coordinated with appropriate agencies, is recommended.