

AD-778 467

PRIMITIVE MODELS FOR COMPUTER SECURITY

CASE WESTERN RESERVE UNIVERSITY

PREPARED FOR
AIR FORCE SYSTEMS COMMAND
ELECTRONIC SYSTEMS DIVISION

23 JANUARY 1974

DISTRIBUTED BY:

NTIS

National Technical Information Service
U. S. DEPARTMENT OF COMMERCE

CLASSIFICATION	SECRET	<input checked="" type="checkbox"/>
BY	DISPOSITION AVAILABILITY CODES	
DATE	AVAILABILITY OF SPECIAL	
A		

LEGAL NOTICE

When U. S. Government drawings, specifications or other data are used for any purpose other than a definitely related government procurement operation, the government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

OTHER NOTICES

Do not return this copy. Retain or destroy.

REVIEW AND APPROVAL

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

William R. Price
 WILLIAM R. PRICE, Lt, USAF
 Computer Security Branch

Roger R. Schell
 ROGER R. SCHELL, Major, USAF
 Chief, Computer Security Branch

FOR THE COMMANDER

P.R. Veckery
 FOR ROBERT W. O'KEEFE, Colonel, USAF
 Director, Information Systems Technology
 Deputy for Command & Management Systems

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ESD-TR-74-117	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) PRIMITIVE MODELS FOR COMPUTER SECURITY		5. TYPE OF REPORT & PERIOD COVERED
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) K. G. Walter W. F. Ogden W. C. Rounds, et al		8. CONTRACT OR GRANT NUMBER(s) F19628-73-C-0185
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Computing and Information Sciences Case Western Reserve University Cleveland, OH 44106		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 5550
11. CONTROLLING OFFICE NAME AND ADDRESS Deputy for Command and Management Systems Hq Electronic Systems Division (AFSC) L G Hanscom Field, Bedford, MA 01730		12. REPORT DATE 23 January 1974
		13. NUMBER OF PAGES 37
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A
15. 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		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Security Models Computer Security Kernel		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) This report presents a mathematical model which specifies the security constraints applicable to computer systems which simultaneously handle data of different sensitivity levels. This model is used to develop a model of security for computer systems which have directory structured file systems. Reproduced by NATIONAL TECHNICAL INFORMATION SERVICE U S Department of Commerce Springfield VA 22151		

Interim Technical Report

PRIMITIVE MODELS FOR COMPUTER SECURITY

By K. G. Walter, W. F. Ogden, W. C. Rounds, F. T. Bradshaw, S. R. Ames,
and D. G. Shumway

Project: Abstract Model for Computer Security, Contract No. F19628-
73-C-0185, ESD, USAF, Hanscom Field, Bedford, Mass. 01730

Department of Computing and Information Sciences, Case Western Reserve
University, Cleveland, Ohio 44106

Dated: 23 January 1974

1-A

ACKNOWLEDGMENTS

We are grateful to Major Roger Schell and Lieutenant Paul Karger of the Air Force Electronic Systems Division and to James P. Anderson for supplying background information about this subject and for providing frequent comment and criticism during the development of these models. We are also grateful to Leroy Smith, Richard Rhode, and C. Chandrasekaran of the MITRE Corporation for their detailed critique of the first version of this report.

FOREWORD

This document reports on a continuing research effort in secure computer systems undertaken by Case Western Reserve University under Contract No. F19628-73-C-0185.

This report attempts to specify requirements for a secure computer system based upon the development and verification of a mathematical model. The Air Force has required the contractor to model the military security system by providing simple and effective security controls and has discouraged attempts to provide broad, general security controls which directly address such issues as "mutually suspicious subsystems."

This report identifies sufficient characteristics for a file system within a multilevel secure computer system. ESD has already applied these results to enhance the file system design of the Multics system recently acquired by the Air Force Data Services Center.

TABLE OF CONTENTS

	Page
ACKNOWLEDGMENTS.....	ii
FOREWORD.....	iii
1. GENERAL INTRODUCTION.....	1
1.1 A Brief Discussion of the Problem.....	1
1.2 Scope of the Project.....	1
1.3 Some Problems Not Being Addressed.....	2
1.4 The Proposed Method of Attack.....	2
1.5 General Methodology.....	3
1.6 An Outline of the Report.....	4
2. THE BASIC MODEL.....	5
2.1 Introduction.....	5
2.2 Basic Elements, Functions, and Relations.....	5
2.3 Axioms.....	10
2.4 The Basic Security Theorem.....	10
2.5 The Air Force Security Lattice.....	12
2.6 A Hypothetical Need-to-Know Clearance Lattice.....	14
3. A DIRECTORY STRUCTURE MODEL.....	17
3.1 Introduction.....	17
3.2 The Basic Objects, Functions and Relations.....	17
3.3 Axioms.....	19
3.4 Implications of the Axioms.....	20
4. CONCLUSION.....	24
4.1 Summary.....	24
4.2 Future Topics.....	25
REFERENCES.....	26
APPENDIX A: ALTERNATE FILES SYSTEMS.....	27
A.1 Directory Trees as Repositories.....	27
A.2 Uncleared Directories.....	28
NOTATION.....	29

1. GENERAL INTRODUCTION

1.1 A Brief Discussion of the Problem

As increasing amounts of data are committed to large on-line data bases, a problem of increasing importance and complexity is that of ensuring security of the data. In general, an attack on the data security problem is made more difficult by the lack of a universally-accepted definition of security. In this project, the particular objective is the introduction of the governmental security scheme into a multi-programmed computer environment. (See Preliminary Notes on the Design of Secure Military Systems in [9]).

Initial investigations indicate that current computer hardware designs will not require major changes to accommodate adequate security provisions. Hence, the first objective is to develop a security system which will supplement systems similar to Multics as implemented on the Honeywell 6180.

1.2 Scope of the Project

Computer security, particularly in the military environment, has many components -- physical security, user authentication, etc. Various of these components are being addressed in a number of current projects.

The objectives of this project are (i) to develop a model which imposes the governmental security scheme on an open-computing environment and (ii) to investigate the security properties and ramifications of such a model. In this context, 'open environment' has two meanings (Anderson [1, Vol. II, p. 4]). First, we mean a situation where not all system users are cleared for the classification of information being processed on the system. Second, we mean that users are able to program in assembly language or any common higher-level language. Currently, these situations create unacceptable security hazards in existing systems.

1.3 Some Problems Not Being Addressed

As indicated above, some aspects of the overall computer security problem will not be addressed in this project. For example, we assume that some external agent is responsible for the assignment of security clearances to individual users and we do not address the problem of providing physical security to objects such as terminals, transmission lines, tapes, cards, disks, etc.

The important problem of user authentication at log-on is not addressed in this project. (This problem is another project assigned to ESD/DCW.) We also have no intention of attempting to verify the correctness of all available software on the system, although we shall be concerned with controlling the potential damage caused by faulty routines.

1.4 The Proposed Method of Attack

An essential aspect of computer system development and implementation is the conceptualization of system structure. The structure proposed for this development is a hierarchy of layers, hardware and/or software, each of which provides one or more machines, virtual or real, to higher layers and external system users [2]. From the perspective of the operating system and users, the lowest layer contains a hardware/software subsystem called the security kernel, which handles the basic protection and security on the system. Higher layers, supported by the kernel, will make the system more convenient for the user.

In specifying and implementing the security kernel, two conflicting requirements must be resolved. On one hand, the kernel should be as small as possible, since the correctness of its functions must be carefully checked. However, to provide adequate security, it appears that many subsystems such as I/O handlers must be included in the kernel, greatly enlarging its size. This increases significantly the difficulty of verifying the kernel; in fact, the feasibility of doing so becomes extremely dependent on the structure of the kernel.

1.5 General Methodology

Initially the levels of abstraction approach to the development of a security kernel seemed more promising than other alternatives. The methodology involves the stepwise refinement of computational tasks and has proved effective in the construction of algorithms and subsystems to perform specific functions. However, a closer inspection of the basic assumptions of stepwise refinement or levels of abstraction methodology indicated several difficulties in its straightforward application to the computer system security problem.

We can regard the stepwise refinement technique of algorithm (system) construction as a methodology for the organization of design decisions. A fundamental assumption of the stepwise refinement methodology is that the design decisions are directed toward a well-defined goal and are restricted by a clearly specified set of constraints. That is, in the construction of an algorithm by the stepwise refinement technique, the intended function of the algorithm is assumed to be well-defined. Moreover, the decision steps are guided by constraints such as the set of available operations on the machine or in the programming language.

Precisely these basic assumptions were not satisfied in the case of computer system security. While a well-accepted statement of the military security system existed, a precise and consistent interpretation in a computer system application did not. Furthermore, the constraints of the computer security problem were not well-understood, e.g., the security implications of certain operations on user data were unclear.

Consequently, the initial phase of the project consisted of specifying precisely the goal and constraints of computer system security in the context of the military security requirements.

The approach taken has been to develop uninterpreted abstract models of security. This allows us to extract the most important features of a given situation and to obtain results which apply to other similar situations. Our first model gives a definition of

security which applies to a wide variety of situations.

The models developed in this project are not independent. At the most abstract level, a set of models address different aspects of the problem. Subsequent, more specific models will refine and possibly combine one or more previous models. Hence, later models will be possible interpretations of previous models.

The goal of this approach is to develop a model which describes in detail the security kernel in a secure computing system. The constraints which guide this approach are provided (i) by the definition of security given in the most abstract models and (ii) by implementations and hardware considerations.

The success of this model refinement technique will have several clear benefits. We will have defined a model which, if correctly implemented, will ensure secure behavior according to a precise definition of security. In addition, this model will provide a precise definition of the goal and constraints for a stepwise implementation of a security kernel.

1.6 An Outline of the Report

This report presents versions of the two models which we have developed so far. The model described in Chapter 2 formalizes the intended effect of the military clearance/classification system.

It models the security system as a set of abstract entities: information repositories, each having some given security class, and agents, also having security classes. The agents may observe and modify various of the repositories. After some formalization, the fundamental result of Chapter 2 is given in a Basic Security Theorem. This theorem states that if the security axioms are enforced, then no information can be transferred to an unsafe place. The limitation of the Chapter 2 security model is that it is static -- not allowing for the creation or deletion of repositories or agents.

Chapter 3 then introduces the concept of a directory system on the Chapter 2 model. By introducing Files and Directories as repositories

this model becomes less abstract than that of the Chapter 2 model. As the file system proposed in Chapter 3 is not the only conceivable design, Appendix A explores two alternate file systems which we investigated.

2. THE BASIC MODEL

2.1 Introduction

The objective of this chapter is to develop a general model for an information processing system which is "secure in that it allows access to information only as defined by rules governing the dispersal of classified information" [7]. The resultant model will serve as a definition of mandatory security in subsequent chapters.

Design requirements are to provide a "military time-sharing system operating in multilevel security mode" [9, p. IV-28]. That is, "a mode of operation under an operating system ... which provides a capability permitting various levels and categories or compartments of material to be concurrently stored and processed in an ADP system. In a remotely accessed resource-sharing system, the material can be selectively accessed and manipulated from variously-controlled terminals by personnel having different security clearances and access approvals..." (DOD 5200:28-M as quoted by Downey) [9, p. IV-28]. The system must be secure in the sense that it permits no "unauthorized disclosure of information" [5, p. 14].

The pioneering work of Lampson [4] was followed in the partitioning of the information store into a set of disjoint repositories (objects) and the use of a set of agents (domains or subjects) for the processing and transferring of the stored information. The repositories are passive information-storing elements in the system. The information stored in a repository can only be modified by an agent, and that modification can only be detected by another agent through an observation. Information is transferred from one repository to another by an observation of the first repository by an agent and the modification of the second by the same agent. The modification is expected to

reflect the information stored in the first repository by an appropriate change in the information stored in the second. Since all transfers are from repository to repository through an agent, it is possible to control information flow in the system by controlling the observation and modification privileges of each of the agents.

The repository is the smallest unit of information storage to which observation and modification access is controlled. All information stored in a repository is thus considered to be of uniform sensitivity. As a measure of this level of sensitivity, a security class is associated with each repository. There may be several repositories with the same security class. The set of repositories is thus subdivided into disjoint subsets of repositories with equal security class which will therefore have the same mandatory access restrictions.

Rather than keeping lists specifying which agents can access which subset of repositories, we associate a security class with each agent. The set of security classes then becomes a common measure which makes the set of agents and set of repositories comparable and which can be used to determine what observations and modifications are to be permitted.

For motivation of the method of solution and of the terminology, let us consider the Air Force regulations and procedure for preventing compromise of classified information. For this purpose, repositories are documents, and agents are personnel. The security class of a document is its classification, and the security class of an individual is his clearance. An individual may read (observe) a document only if its classification is less than or equal to his clearance. Notice that this is a necessary but not a sufficient condition. It gives a basis for controlling observations, but there must also be a basis for determining which modifications should be allowed. Recall that our objective is to prevent unauthorized disclosure. Information must not be transferred to a repository with security class lower than that of the source repository. In order to prevent any agent from transferring information to a repository with insufficient security class in the model, an agent will not be allowed to modify a repository

unless its security class is greater than or equal to that of the agent.

In the preceding paragraphs there is a relation on the set of security classes implied by the comparisons between elements. Let us consider what properties the relation must have. Certainly an agent with given security class (clearance) should be allowed by the mandatory security system to observe or modify any repository with the same security class (classification). Then since every security class must be less than or equal to itself, the relation must be reflexive. In order to control possible information transfer from one repository to another through one or more intermediate repositories, it is necessary that the relation be transitive.

According to Popek [8, Chap. 5] in his general treatment of access maps using restriction graphs, the relation on the set of security classes can be a partial ordering. However, antisymmetry is not essential for the basic theorem of this chapter. It is sufficient to assume that the relation on the set of security classes is a pre-ordering.

In contrast to Schiller [8] we believe that what he calls the "control structure" of the system is integrally related to the "information structure," and that penetration attacks on this control structure must be prevented by applying certain restraints in the model rather than by giving the system designer the "responsibility to systematically determine all possible channels through the control structure in his system, and whether or not the transmission rate from their use in a penetration is acceptable" [10, p 7].

Conceptually, a repository is any object in the system which can be modified in such a way that this modification can be subsequently observed. Each process in a computer should be interpreted to be an agent; however, since the state of a process (running, blocked, ready, etc.) is information which can be modified and observed by other system elements, a process must also be considered to be a repository, and, accordingly, its state information must be protected by the security kernel. The proposed model is intended to be sufficiently general to include all possible information channels. We believe that to leave large classes of potential information channels to be

systematically discovered and dealt with by the system designer is to circumvent the very idea of design through mathematical models.

2.2 Basic Elements, Functions, and Relations

We now introduce some notation and formalize the ideas discussed in the preceding section.

The model for mandatory security in an information processing system is an 8-tuple:

$$M_0 = (R, A, C, \theta, \mu, \preceq, Cls, Clr)$$

where

R is a set of repositories.

A is a set of agents.

C is a set of security classes.

$\theta \subseteq A \times R$ is the "observe" relation. ($\underline{a} \theta \underline{r}$ means that agent \underline{a} can observe the information stored in repository \underline{r} .)

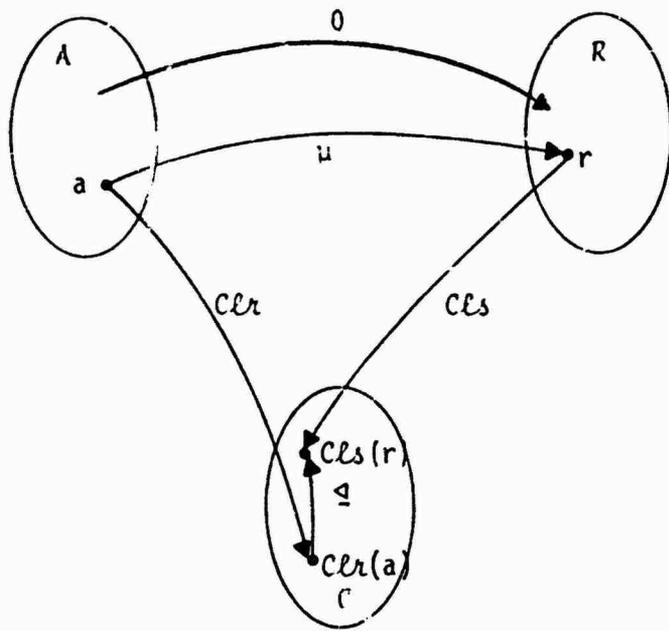
$\mu \subseteq A \times R$ is the "modify" relation. ($\underline{a} \mu \underline{r}$ means that agent \underline{a} can modify the information stored in repository \underline{r} .)

$\preceq \subseteq C \times C$ is a pre-ordering of the set of security classes.

$Cls: R \rightarrow C$ is the "classification" function which associates a security class with each repository. (Informally $Cls(\underline{r})$ will be referred to as the classification of repository \underline{r} .)

$Clr: A \rightarrow C$ is the "clearance" function which associates a security class with each agent. (Here again $Clr(\underline{a})$ will be referred to as the clearance of agent \underline{a} .)

The model M_0 can be illustrated by the following picture:



2.3 Axioms

The model M_0 has four axioms. The first two simply state explicitly that the relation \leq is a pre-ordering of the set C of security classes.

A1: For all $c \in C$, $c \leq c$. (\leq is reflexive.)

A2: For all $c, d, e \in C$, $c \leq d$ and $d \leq e$ implies $c \leq e$. (\leq is transitive.)

The second two axioms state the conditions necessary for observation and modification by agents.

A3: For all $a \in A$ and $r \in R$, $a \theta r$ implies $Cls(r) \leq Clr(a)$.
(That is, if agent \underline{a} can observe repository \underline{r} , then the clearance of \underline{a} must be greater than or equal to the classification of \underline{r} .)

A4: For all $a \in A$ and $r \in R$, $a \mu r$ implies $Clr(a) \leq Cls(r)$.
(That is, if an agent \underline{a} can modify repository \underline{r} , then the clearance of \underline{a} is less than or equal to the classification of \underline{r} . Agent \underline{a} can modify only those repositories with equal or higher security class.)

2.4 The Basic Security Theorem

Using these four axioms, we now prove that in M_0 no information can ever be transferred to a repository in which it can be observed by an agent that does not have sufficient clearance to observe the source repository.

In preparation for the statement and proof of the basic security theorem about M_0 , we make the following definitions.

Define the "transfer" relation $\tau \subseteq R \times R$ by $\underline{r} \tau \underline{s}$ if and only if there is an agent \underline{a} such that $\underline{a} \theta \underline{r}$ and $\underline{a} \mu \underline{s}$, where \underline{r} and \underline{s} are repositories. Thus $\underline{r} \tau \underline{s}$ means that there is an agent which can transfer information from repository \underline{r} to repository \underline{s} . It is actually

the transitive, reflexive closure τ^* of τ which is needed in the theorem, since $\underline{r} \tau^* \underline{s}$ means that information can eventually be transferred from \underline{r} to \underline{s} . In this case there is a finite sequence of repositories $\{\underline{r}_i\}$ such that $\underline{r} = \underline{r}_1$, $\underline{s} = \underline{r}_{n+1}$, and $\underline{r}_i \tau \underline{r}_{i+1}$ for all i , $1 \leq i \leq n$.

Further, if $\underline{r} \tau^* \underline{s}$, then we say there is an information transfer path from repository \underline{r} to repository \underline{s} .

THEOREM: If there is an information transfer path from repository \underline{r} to repository \underline{s} in M_0 , then $Cls(\underline{r}) \trianglelefteq Cls(\underline{s})$.

PROOF: By definition, if there is an information path from repository \underline{r} to repository \underline{s} , then $\underline{r} \tau^* \underline{s}$. We first establish that $\underline{r} \tau \underline{s}$ implies $Cls(\underline{r}) \trianglelefteq Cls(\underline{s})$. For if $\underline{r} \tau \underline{s}$ then there is an agent \underline{a} such that $\underline{a} \theta \underline{r}$ and $\underline{a} \mu \underline{s}$. By axioms A3 and A4, $Cls(\underline{r}) \trianglelefteq Clr(\underline{a})$ and $Clr(\underline{a}) \trianglelefteq Cls(\underline{s})$. By transitivity of \trianglelefteq , $Cls(\underline{r}) \trianglelefteq Cls(\underline{s})$.

Now define a new relation $\lambda \subseteq R \times R$ by $\underline{r} \lambda \underline{s}$ if and only if $Cls(\underline{r}) \trianglelefteq Cls(\underline{s})$. (That is, $\underline{r} \lambda \underline{s}$ means the security class of \underline{r} is less than or equal to the security class of \underline{s} .)

Notice that λ is a reflexive and transitive relation. For any $\underline{r} \in R$, axiom A1 states that $Cls(\underline{r}) \trianglelefteq Cls(\underline{r})$, and so $\underline{r} \lambda \underline{r}$ (i.e., λ is reflexive) by the definition of the relation λ . If $\underline{r}, \underline{s}, \underline{t} \in R$, such that $\underline{r} \lambda \underline{s}$ and $\underline{s} \lambda \underline{t}$, then, by the definition of λ , $Cls(\underline{r}) \trianglelefteq Cls(\underline{s})$ and $Cls(\underline{s}) \trianglelefteq Cls(\underline{t})$. By axiom A2, $Cls(\underline{r}) \trianglelefteq Cls(\underline{t})$, and, again by definition of λ , $\underline{r} \lambda \underline{t}$ (i.e., λ is transitive).

The relation λ contains the relation τ , since $\underline{r} \tau \underline{s}$ implies $Cls(\underline{r}) \trianglelefteq Cls(\underline{s})$, which implies $\underline{r} \lambda \underline{s}$. Recall that by definition the relation τ^* is the minimal transitive, reflexive relation containing the relation τ . It follows that the relation τ^* is contained in the relation λ . This proves the theorem, since for $\underline{r}, \underline{s} \in R$, $\underline{r} \tau^* \underline{s}$ implies $\underline{r} \lambda \underline{s}$, which implies $Cls(\underline{r}) \trianglelefteq Cls(\underline{s})$.

2.5 The Air Force Security Lattice

The applicability of M_0 to a system which enforces the Air Force clearance/classification and compartmentalization restrictions will be demonstrated by showing that these two schemes can be combined to give a single set of security classes. Air Force "need-to-know" restrictions have not been included because, as will be shown in the next section (2.6), strict mandatory enforcement of need-to-know gives a system of limited usefulness, and it does not model the actual military use of need-to-know.

In a computer system application, need-to-know will be implemented through access control lists. The system cannot be expected to decide which names and access rights should be put on the access control lists, but it must be certified that the lists are implemented and maintained correctly. It is more realistic in this instance to allow the creator/owner of a repository (file) or the project director to decide who shall have access to a given repository. The user gives access rights at his own discretion, with full confidence that no programming error or oversight will defeat the mandatory enforcement of clearance/classification and compartmentalization restrictions by the system kernel.

See Downey's discussion of the three dimensions of military security and user responsibility [9, Sec. 3.1, p. IV-28]. Notice that our model does not give the user as much responsibility as Downey suggests. In our model if a user has observe access to a repository, he may do whatever he pleases with the information stored there; however, he may not use the facilities of the system to pass it to an unauthorized user, nor can an unknown program bug divulge it. The user is also ensured that, while a borrowed program may destroy or change information to which he has modify access, it will not transfer information to any repository where it would be observable by anyone with lower security class. (A malicious borrowed program is often called a "Trojan Horse." [1, Vol. II, p. 50]. Since the system cannot appraise user intentions, it prohibits intentional as well as unintentional unauthorized disclosures.

Let us now write out the details of the security lattice which describes the Air Force classification system. Let S_{en} be the set of

sensitivity levels (i.e., unclassified, confidential, secret, etc.). Sen is a lattice because it is linearly ordered.

Let Cmp be the set of compartments of subject matter (China, nuclear, etc.). Generally the information stored in a given repository may be included in more than one compartment, hence the component of a security class concerned with compartmentalization will actually be a subset of compartments to which the information belongs. Although all possible subsets of Cmp may not be needed in practice, our formal treatment will use the entire power set $P(Cmp)$ of Cmp . $P(Cmp)$ is a lattice naturally ordered by set inclusion. The two lattices Sen and $P(Cmp)$ can be combined to form the product $Sen \times P(Cmp)$ which is itself a lattice (MacLane and Birkhoff [6, p. 489]) with order relation defined as follows: for $(c,D), (e,F) \in Sen \times P(Cmp)$, $(c,D) \leq (e,F)$ if and only if $c \leq e$ in Sen and $D \subseteq F$ in $P(Cmp)$. Thus an agent may observe a repository only if the classification level of the repository is less than or equal to the clearance level of the agent and the set of compartments associated with the repository is a subset of the set of compartments associated with the agent.

Since the set $C_0 = Sen \times P(Cmp)$ is a lattice, under the ordering \leq defined above, axiom A1 and A2 will be satisfied. Being a lattice, C_0 has stronger properties than required for mandatory security as defined by M_0 . While not necessary, those additional properties may be useful in practice. For example, given any two classes in C_0 , there is a minimal class which is greater than or equal to either one. Also, C_0 has a greatest and a least element.

The basic theorem then states that, in a system which uses the Air Force security lattice C_0 to restrict the observe and modify relations according to axioms A3 and A4, there can be no transfer of information from a repository with one sensitivity and set of compartments to a repository with lower sensitivity or smaller set of compartments. And, since the only access to repositories is through agents, there can be no unauthorized disclosure of information.

The modeled system is quite similar to the real world except for axiom A4 which states that an agent may not modify a repository with lower security class. Under that restriction, if agents are acting

on behalf of system users, a user who has, say, secret clearance could not send any information to an uncleared user through the system. This is necessary since the system cannot interpret which information is classified and which is not.

To realistically apply the basic theorem to the Air Force security procedure, we might suggest that a person be allowed to operate as an agent with any clearance up to and including his actual clearance. This places the responsibility on the user to decide at which level he should operate. In making the decision to operate at a reduced clearance, the user relinquishes the right to observe any material classified higher than his "working level." In this way axiom A4 can be satisfied and the result of the theorem ensured.

2.6 A Hypothetical Need-To-Know Clearance Lattice

The Air Force further ensures the privacy and integrity of information by requiring that, to access information, one must not only have proper clearance but must also have established a "need-to-know" for the information. The fact that Jones is cleared to see material in a given security class does not mean he can read a document with that classification written by Smith, unless he has been extended the proper need-to-know authorization. In attempting to describe this need-to-know security scheme, it is natural to try to fit it into the basic security model M_0 of Section 2.3.

When we describe need-to-know security in terms of M_0 , we interpret the set C of security classes as being the new set C_1 , described below. Let U be the set of users of the system, and define C_1 to be the power set of users $P(U)$. Define the ordering \triangleleft on C_1 to be reverse set inclusion (i.e., $B \triangleleft D$ if and only if $D \subseteq B$). This relation is reflexive, since for all $B \in P(U)$, $B \subseteq B$. The relation \triangleleft is also transitive since if $B \triangleleft D$ and $D \triangleleft F$, then $D \subseteq B$ and $F \subseteq D$, so by the transitivity of \subseteq , $F \subseteq B$ (which states that $B \triangleleft F$). Accordingly, \triangleleft is a preordering of C_1 , as required by axioms A1 and A2.

In this example, the classification function cls will assign to each repository r a set of users $cls(r)$ who are allowed to observe the

contents of repository r (i.e., who have the "need-to-know" authorization for repository r).

The clearance function Clr will assign to each agent a a set of users $Clr(a)$ who are represented by the agent a . Now axiom A3 says that, in order for agent a to observe repository r , $Cls(r) \subseteq Clr(a)$. In other words, the set of users who are represented by agent a are a subset of the users who are allowed to observe repository r .

Axiom A4, on the other hand, says that, in order for an agent a to modify a repository r , the set of users whom the agent a represents must include the set of users who can observe repository r . When we apply the security theorem of Section 2.4 to this example, we see that information cannot be passed to repositories which larger sets of users can observe. For example, it is impossible to pass the information in a repository to a user who does not already have access to the information in the repository, since this amounts to "declassifying" this information.

Unfortunately, this security scheme is much too rigid to be of any use, except perhaps in the special case of a small environment. As we have just seen, it is impossible to straightforwardly give a new user Jones access to the contents of a repository r which was created before it was decided that Jones had a "need-to-know" the contents of r . One way around this difficulty might be to recreate the contents of r in a new repository r' which includes Jones in its "need-to-know" classification. This process would involve accessing all the repositories which we used when we initially created the contents of r . If Jones is not "need-to-know" classified to see some of these repositories, they will also have to be reconstructed. The process of reconstruction would continue until we reach a collection of repositories, all of which Jones has access to. This could certainly be a costly and cumbersome process.

An alternative approach would be to have a security officer authorize the direct transfer of the contents of repository r to repository r' . However, if repository r is frequently updated, this approach will require frequent actions by the security officer, and this really circumvents the mandatory security system. It is natural then to direct

all subsequent updates to repository r' rather than repository r . This means that all agents which modify r' will have to include Jones among the users they represent, and this means that Jones will have to be included in the classification of all repositories which these agents observe while updating r' . Accordingly, all of these repositories will have to be "declassified" by the security officer. This leads to the same proliferation problem which we saw previously.

Because of these problems, we are led to introduce a less strict "need-to-know" security system in which the responsibility for protecting the contents of repositories is left to the users. We will refer to this system as the discretionary security system.

This system involves attaching to each repository a list of users who are allowed to observe the contents of that repository. There is a major disadvantage to this system: if we let user Smith observe one of our repositories, we cannot be certain that he will not pass the contents along to user Jones. This is similar, however, to the real world situation.

An advantage of the discretionary security system is that we only have to check whether one user is on a need-to-know list rather than having to check whether one list of users is contained in another list of users. Checking for individual membership is usually much faster than checking for containment.

There is another security problem which the discretionary security system should confront, and that is the problem of sabotage. We could construct a strict "need-to-modify" security system to deal with this problem, but it would be just as cumbersome as the strict "need-to-know" system. The proposed discretionary security system will deal with this problem by attaching to each repository a list of users who are allowed to modify the contents of that repository.

Since the discretionary security is not as strict about controlling access to repositories as the mandatory security is, we will provide the user with additional mechanisms for controlling his agents. These will take the form of agent privileges which the user may selectively revoke.

In this chapter it was our intention to define security in terms

of a mandatory scheme involving security classes. When we tried to apply this sort of security scheme to need-to-know security, we were led to the idea of a discretionary security system which the users will apply. This discretionary security is one of the topics to be considered in future work.

The next chapter pursues the interaction between directory structures and mandatory security.

3. A DIRECTORY STRUCTURE MODEL

3.1 Introduction

Our approach to building secure computer systems involves describing a variety of models which capture different aspects of the problem in varying degrees of detail. In this chapter, we will be concerned with a model which adds additional structure to the set of repositories discussed in Section 2 of the preceding chapter. We will explore the additional restrictions which this structure and the security axioms A3 and A4 imply.

Most large computer systems use a directory scheme to structure their file systems, so we will model this sort of repository structure. Certain computer applications such as question answering systems require more complex data structures, and it will eventually be important to explore the security requirements of such systems. A brief study of more complex data structures indicates that they require considerably more complex security restrictions, so, for the moment, we will confine our attention to directory structures.

3.2 The Basic Objects, Functions and Relations

Formally, this model concerns three sets, two functions and four relations. Several of these concepts were introduced in the model in the preceding chapter.

A = a set of agents.

C = a set of security classes.

F = a tree of files (directories and segments).

$Clr: A \rightarrow C$ is a clearance function for agents.

$Cls: F \rightarrow C$ is a classification function for files.

$\preceq \subseteq C \times C$ is the preordering of the security classes.

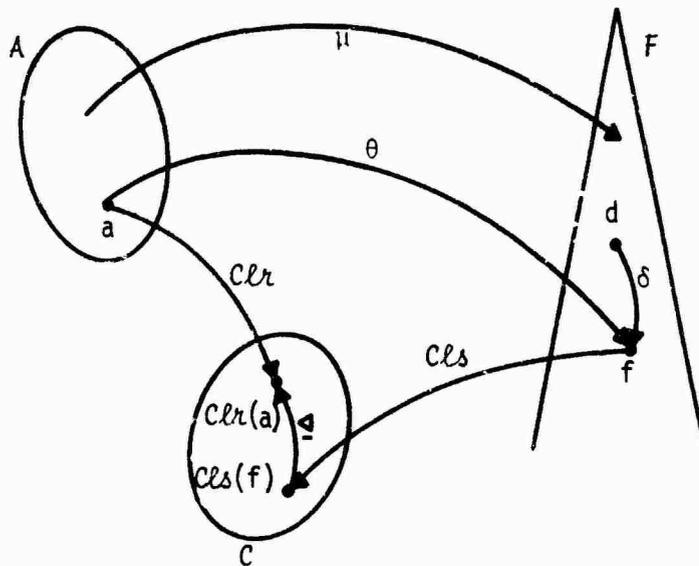
$\theta \subseteq A \times F$ is the "can observe" relation between agents and files.

$\mu \subseteq A \times F$ is the "can modify" relation between agents and files.

$\delta \subseteq F \times F$ is the "dominates" relation between files. (This relation will be used to indicate that the files form a tree.)

The tree structure of the files is a reflection of the directory naming scheme for addressing segments. Each directory contains pointers to other directories and segments which it directly dominates.

The following picture illustrates the various sets, functions, and relations involved in this model:



Now we will discuss the axioms which formally describe this model.

3.3 Axioms

The first four axioms are retained from the preceding chapter.

- Ax1: For all security classes c in C , $c \leq c$ (\leq is reflexive).
- Ax2: For any security classes c_1, c_2, c_3 in C , if $c_1 \leq c_2$ and $c_2 \leq c_3$, then $c_1 \leq c_3$ (\leq is transitive).
- Ax3: For any agent a in A and file f in F , $a \theta f$ implies that $Cls(f) \leq Clr(a)$. (Agents can only observe files of classification less than or equal to their clearance.)
- Ax4: For any agent a in A and file f in F , $a \mu f$ implies that $Clr(a) \leq Cls(f)$. (Agents can only modify files of equal or higher classification.)

The next four axioms formalize the tree structure of the set of files.

- Ax5: For all files f in F , $f \delta f$ (δ is reflexive).
- Ax6: For any files f and g in F , if $f \delta g$ and $g \delta f$, then $f = g$ (δ is antisymmetric).
- Ax7: For any files f, g , and h in F , if $f \delta g$ and $g \delta h$, then $f \delta h$ (δ is transitive).
- Ax8: For any files f, g , and h in F , if $g \delta f$ and $h \delta f$, then $g \delta h$ or $h \delta g$ (δ has the tree property).

The next axiom reflects the fact that in a directory system, when a directory is deleted, all the files below that directory become inaccessible.

Ax9: For any agent a in A and any files f and g in F , if $a \theta g$ and $f \delta g$, then $a \theta f$. (If an agent can observe a file, then it can observe any directory which dominates the file.)

This axiom recognizes certain implicit information paths which result from the nature of the directory system. To illustrate the difficulty, suppose that an agent a' deletes a directory f which dominates a file g , then another agent a can tell that the file g is no longer observable. This constitutes a potential communication path between a' and a . In effect, a can make a limited observation of directory f by determining whether file g is still observable. (See Lampson [3] for a discussion of similar problems.)

Axiom Ax10 is introduced, because, in most computer systems, when an individual tries to write on a file f which he is not allowed to access, he gets an error message. The presence or absence of this error message constitutes an observation of the file f .

Ax10: For all agents a in A and files f in F , if $a \mu f$, then $a \theta f$. (An agent can observe all files which he can modify.)

It would certainly be possible to build systems in which no information would be given regarding whether or not a modify command was successfully carried out, but such systems would be unpleasant to use. A more sophisticated system might not return information about the success of modifications of files when the clearance of the agent was strictly less than the classification of the file.

3.4 Implication of the Axioms

In this section we will explore some of the consequences of the axioms introduced in the preceding section. Some of these consequences lead us to introduce another axiom.

To begin with, we see that axioms Ax9 and Ax3 combine to give us

Proposition 1: For any agent a in A and any files f and g in F , if $a \theta g$ and $f \delta g$, then $Cls(f) \leq Clr(a)$.
(An agent must have a clearance which is greater than or equal to the classification of any directory in which it observes a file.)

Axiom Ax9 says that, if $a \theta g$ and $f \delta g$, then $a \theta f$. Axiom Ax3 says that, if $a \theta f$, then $Cls(f) \leq Clr(a)$, so the proposition follows immediately.

In order to state the next proposition concisely, we introduce the equivalence relation \equiv on the security classes C which is naturally defined from the preordering \leq by

$$c_1 \equiv c_2, \text{ if and only if } c_1 \leq c_2 \text{ and } c_2 \leq c_1.$$

Axiom Ax10 then leads to

Proposition 2: For all agents a in A and files f in F , if $a \mu f$, then $Clr(a) \equiv Cls(f)$. (Agents can only modify files whose classification is equivalent to their clearance.)

First note that axiom Ax4 says that, if $a \mu f$, then $Clr(a) \leq Cls(f)$. Now axiom Ax10 states that if $a \mu f$, then $a \theta f$, and axiom Ax3 states that if $a \theta f$, then $Cls(f) \leq Clr(a)$. So, by the definition of \equiv , $Clr(a) \equiv Cls(f)$.

Proposition 1, axiom Ax10, and the first four axioms have the following consequence.

Proposition 3: For any agent a in A and any files f and g in F , if $a \mu g$ and $f \delta g$, then $Cls(f) \leq Cls(g)$.

(A file which can be modified by some agent must have a classification which is greater than or equal to the classification of any directory which eventually contains it.)

Axiom Ax4 says that, if $a \mu g$, then $Clr(a) \leq Cls(g)$. Now note that Ax10 asserts that, if $a \mu g$, then $a \theta g$. So we can invoke Proposition 1 which says that, if $a \theta g$ and $f \delta g$, then $Cls(f) \leq Clr(a)$. Using the transitivity of \leq (Ax2), $Cls(f) \leq Clr(a) \leq Cls(g)$ implies $Cls(f) \leq Cls(g)$ as desired.

The implications of Proposition 3 are clearer if we state it in contrapositive form.

Corollary 1: For any agent a in A and files f and g in F , if $f \delta g$ and $Cls(f) \not\leq Cls(g)$, then not $a \mu g$. (No agent can modify a file g which is contained in a directory whose classification is not less than or equal to the classification of the file g .)

This corollary motivates us to introduce another axiom in order not to have a file system which is cluttered with useless, unmodifiable files.

Ax11: For any files f and g in F , if $f \delta g$, then $Cls(f) \leq Cls(g)$. (Every directory has an equal or lower classification than any file it eventually contains.)

It is interesting to note that we no longer need axiom Ax9 to prove Propositions 1, 2, and 3. Proposition 3, which motivated our new axiom, is an immediate consequence of Ax11. The proof of Proposition 2 did not use Ax9, so only Proposition 1 needs to be reproved.

Proposition 4: Axioms Ax2, Ax3, and Ax11 imply Proposition 1. If we

assume $a \theta g$ and $f \delta g$, then we must prove that $Cls(f) \trianglelefteq Cls(a)$. Axiom Ax11 says that if $f \delta g$, then $Cls(f) \trianglelefteq Cls(g)$. Ax3 says that, if $a \theta g$, then $Cls(g) \trianglelefteq Clr(a)$. So, by the transitivity of \trianglelefteq (Ax2), $Cls(f) \trianglelefteq Cls(g) \trianglelefteq Clr(a)$ implies $Cls(f) \trianglelefteq Clr(a)$.

The axioms and propositions developed in this section will be used to formulate more detailed dynamic models of security systems.

Axiom Ax11 and Propositions 1 and 2 are of particular importance in handling directory structured file systems.

4. CONCLUSION

4.1 Summary

The preceding two chapters set forth the first two models in a projected series of mathematical models whose purpose is to give a systematic presentation of the specification for a secure military computing system. The first model gave an overview of the design objective of the security system. The soundness of this design was established by proving a basic security theorem about it.

The second model concerns the security aspects of classified information in a directory-structured file system. Here, the consequences of certain straight-forward assumptions about the nature of the directory structures -- together with the basic axioms of the first model -- made evident the desirability of imposing additional restrictions on the classifications of directories and segments. This development illustrates one of the hoped-for benefits of formulating a concise mathematical description of the proposed design, namely, that design decisions can be based on logical consequences, rather than just seat-of-the-pants intuition.

Two additional topics were treated as an adjunct to these models. The first was the formulation of a mathematically-precise definition of the governmental classification scheme. The second was a discussion of the inappropriateness of using the general model of security to describe "need to know" type restrictions. Other sorts of mechanisms will have to be introduced into the design at a later stage of development in order to handle "need to know" security properly.

4.2 Future Topics

The two models developed so far are actually static models in that the overall structure of the objects described does not change. However, the computer system for which we are giving specifications will change things over the course of time. Accordingly, at some level of detail dynamic models must be introduced. In particular, the model at the next level of detail will be an automaton that reflects the possible changes of security state which our system can go through. In addition, the various security related functions which processes on the system can perform must be specified.

Chapter 2 did not resolve the problem of finding an adequate discretionary security system to handle "need to know" type security. Providing this type of security is very important both because it prevents information from being widely disseminated on a given security level and hence being more vulnerable to compromise and also because it prevents over-classification of information by allowing individuals to work on material at a low classification level with full confidence that it will not be compromised by the system.

In addition to providing a complete description of the full set of operations to be provided by the virtual machine which constitutes the security system, this project should provide descriptions of the virtual machines which are the various levels of abstraction necessary to implement this security system on the hardware which is available.

A long-range objective of this project is to discover general techniques which will allow an even more systematic design of future security systems. This objective makes it imperative to develop this particular design in as clearly structured a way as possible, so that the underlying principles can be discovered.

REFERENCES

- [1] Anderson, James P., "Computer Security Technology Planning Study," Vols. I and II, ESD-TR-73-51, Electronics Systems Division/Air Force Systems Command, L. G. Hanscom Field, Bedford, Mass., October 1972.
- [2] Dijkstra, E. W., "Notes on Structured Programming," Ch. 1, Structured Programming, Dahi, Dijkstra, and Hoare, Academic Press, New York, 1972.
- [3] Lampson, Butler W., "A Note on the Confinement Problem," Communications of the ACM 16, 10 (October 1973), 613-615.
- [4] Lampson, Butler W., "Protection," Proc. Fifth Annual Princeton Conf. on Information Sciences and Systems, Dept. of Elect. Eng., Princeton University, Princeton, New Jersey, March 1971, pp. 437-443.
- [5] LaPadula, Leonard J. and Bell, D. Elliott, "Secure Computer Systems: A Mathematical Model," Vol. II, MTR-2547, The Mitre Corporation, Bedford, Mass., May 1973.
- [6] MacLane, Saunders and Birkhoff, Garrett, Algebra, The Macmillan Company, London, 1967.
- [7] Mitre Corporation, "Secure Operating Systems Design," (Private Communication), The Mitre Corporation, Bedford Mass., 1973.
- [8] Popek, Gerald J., "Access Control Models," ESD-TR-73-106, ESD/AFSC, L. G. Hanscom Field, Bedford Mass., February 1973.
- [9] Schell, Roger R., Downey, P. J., and Popek, G. J., "Preliminary Notes on the Design of Secure Military Computer Systems," ESD-MCI-73-1, L. G. Hanscom Field, Bedford, Mass., June 1973.
- [10] Schilier, W. L., "Design of a Security Kernel for the PDP-11/45," MTR-2709, The Mitre Corporation, Bedford, Mass., June 1973.

APPENDIX A

ALTERNATE FILE SYSTEMS

Introduction

The particular directory system presented in Chapter 3 resulted from making specific choices or assumptions from a set of design alternatives. Other choices or assumptions lead to alternate schemes, and in this Appendix, two such schemes are presented.

A1. Directory Trees as Repositories

From human engineering considerations, it has been suggested that security classes within a directory may inconvenience system users. Hence, in this example, we investigate a multics-like directory system in which there is no distinction of security classes, within a directory.

An immediate consequence is that there must be a different directory tree for each security class. It follows then that a user may no longer have all of his repositories in one directory or even in the same directory tree.

In a directory tree, the repositories (or tree leaves) are, in effect, named by the path from the tree root to the leaf. Then repositories on different directory trees can be in the same position and thus have the same name. In this situation, the only way to distinguish them would be by naming the tree, or giving the security class, and thus the security class becomes a necessary component of every repository specification.

In effect, this approach leads to a single directory tree as does the model in Chapter 3. In this case, however, the root node has different characteristics from other tree nodes. The root node has a fixed number of branches, precisely the number of security classes.

It would appear that this approach does little to reduce user inconvenience.

A2. Uncleared Directories

In this example, the attempt is again made to remove security class checking along paths to a segment by removing security classes from directories, but retaining security classes for segments.

First we must recognize that every object has a security class; "no classification" being the minimum class possible. In this case, that would be the security class of every directory. The directory names constitute some "information," and hence directories must be created by executors with the corresponding minimum clearance.

One must also recognize that the name of an object is a piece of information, disjoint from the object itself. In this example, the question arises of what classification to give to the name of a classified segment. In the Chapter 3 model, the segment "name" had the security class of the appropriate directory. Consequently, this model makes explicit a second form of security checking which is done implicitly by the Chapter 3 model. Regardless of the security class of the information in a segment, the kernel cannot admit the existence of that segment unless the requesting executor has the necessary clearance to know the segment name. It is again possible to have different segments created with the same name, distinguished only by the security class, and security class is a necessary part of a segment name in this example.

Security checking in this system requires more work than the Chapter 3 model with no attendant advantages. In fact, rather than being more convenient, the A.1 and A.2 models appear to be less convenient to the user than the model of Chapter 3.

NOTATION

<u>Symbols</u>	<u>Descriptions</u>	<u>Page Number of Defining Occurrences</u>
M_0	The most primitive security model	8
R	Set of information repositories	8
A	Set of agents	8
C	Set of security classes	8
Cl_s	Classification function for repositories	8
Cl_r	Clearance function for agents	8
\preceq	Pre-ordering of security classes	8
θ	The "can observe" relation between agents and repositories	8
μ	The "can modify" relation between agents and repositories	8
A_1	Security axiom 1	10
A_2	Security axiom 2	10
A_3	Security axiom 3	10
A_4	Security axiom 4	10
τ	The "information can pass" relation between repositories	11
τ^*	The "information can eventually pass" relation	11
λ	The "lower classification" relation on repositories	11
Sen	The set of sensitivity levels	12
Cmp	The set of security compartment	13
C_0	The governmental Security Lattice	13
U	The set of users	14

F	The set of files	18
δ	The "dominates" relation on the set of files	18
Ax1 - Ax11	Directory model Axioms	19-22
\equiv	The equivalence relation on C induced by the partial ordering \leq .	21