LABORATORY FOR
COMPUTER SCIENCE
(formerly Project MAC)

MASSACHUSETTS
INSTITUTE OF
TECHNOLOGY

MIT/LCS/TR-179

NON-DISCRETIONARY ACCESS CONTROL FOR DECENTRALIZED COMPUTING SYSTEMS

Paul A. Karger

This research was supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by the Office of Naval Research under Contract No. N00014-75-C-0661

545 TECHNOLOGY SQUARE, CAMBRIDGE, MASSACHUSETTS 02139

MIT/LCS/TR-179

NON-DISCRETIONARY ACCESS CONTROL FOR DECENTRALIZED COMPUTING SYSTEMS

Paul A. Karger

May 1977

This research was sponsored in part by the Advanced Research Projects Agency (ARPA) of the Department of Defense under ARPA Order No. 2095, which was monitored by the Office of Naval Research under Contract No. N00014-75-C-0661.

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

LABORATORY FOR COMPUTER SCIENCE
(formerly Project MAC)

CAMBRIDGE MASSACHUSETTS 02139

ACKNOWLEDGMENTS

First, I would like to thank my thesis supervisor, Professor J. H. Saltzer, for his guidance and inspiration during the research described in this thesis. Not only did he provide valuable technical insights, but he also provided excellent editorial comments with the rapid turnaround time required by my short deadline for completion.

I must also thank all the members of the Computer Systems Research Group for their comments and suggestions relating to my work. In particular, Dr. David Clark provided several ideas relating to authentication. David Reed's naming scheme was the inspiration for much of chapter 8, and I am grateful to him for allowing me to present his ideas here, prior to their publication. Stephen Kent helped me by critically listening to many of my ideas and pointing out their faults and inconsistencies. I must also thank Mrs. Muriel Webber for her assistance with several of the figures.

I must also thank several individuals from the MITRE Corporation for their advice and comments. Steven Lipner provided suggestions for one-way communication and downgrading and provided valuable editorial comments. Michael Padlipsky and David Snow provided suggestions on end-to-end encryption, and Stanley Ames provided a number of ideas relating to multilevel terminals.

Finally, special thanks must go to Lt. Col. Roger R. Schell of the U. S. Air Force Electronic Systems Division who first introduced me to non-discretionary access controls in 1972, and was the origin and inspiration of many of the ideas presented in this thesis.

This report is based upon a thesis of the same title submitted to the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, on May 12, 1977 in partial fulfillment of the requirements for the degree of Master of Science.

NON-DISCRETIONARY ACCESS CONTROL FOR DECENTRALIZED COMPUTING SYSTEMS

by

Paul Ashley Karger

Submitted to the

Department of Electrical Engineering and Computer Science

on May 12, 1977 in partial fulfillment of the requirements

for the Degree of Master of Science.

ABSTRACT

This thesis examines the issues relating to non-discretionary access controls for decentralized computing systems. Decentralization changes the basic character of a computing system from a set of processes referencing a data base to a set of processes sending and receiving messages. Because messages must be acknowledged, operations that were read-only in a centralized system become read-write operations. As a result, the lattice model of non-discretionary access control, which mediates operations based on read versus read-write considerations, does not allow direct transfer of algorithms from centralized systems to decentralized systems. This thesis develops new mechanisms that comply with the lattice model and provide the necessary functions for effective decentralized computation.

Secure protocols at several different levels are presented in the thesis. At the lowest level, a host to host protocol is shown that allows communication between hosts lacking effective internal security controls as well as hosts with effective internal security controls. Above this level, a host independent naming scheme is presented that allows generic naming of services in a manner consistent with the lattice model. The use of decentralized processing to aid in the downgrading of information is shown in the design of a secure intelligent terminal. Schemes are presented to deal with the decentralized administration of the lattice model, and with the proliferation of access classes as the user community of a decentralized system becomes more diverse. Limitations in the use of end-to-end encryption when used with the lattice model are identified, and a scheme is presented to relax these limitations for broadcast networks. Finally, a scheme is presented for forwarding authentication information between hosts on a network, without transmitting passwords (or their equivalent) over the network.

Thesis Supervisor: Jerome H. Saltzer
Title: Professor of Computer Science and Engineering

TABLE OF CONTENTS

Acknowledgments 2

Abstract 3

Table of Contents 4

List of Figures 7

1. Introduction 9

    1.1 What is a Decentralized Computing System? 10
    1.2 Why Non-Discretionary Access Controls? 11
    1.3 Plan of the Thesis 12

2. Protection Goals for Distributed Systems 13

    2.1 Basic Requirements 13
    2.2 Threats 14
        2.2.1 Threats to Physical Security 14
        2.2.2 Threats to Communications Media 15
        2.2.3 Software Related Threats 17
            2.2.3.1 Direct Attacks 17
            2.2.3.2 Trojan Horse Attacks 19
    2.3 Authentication 20

3. Semantics of Access Control 23

    3.1 Discretionary Access Control 24
    3.2 Non-Discretionary Access Control 26
        3.2.1 Definition of the Lattice Model 27
        3.2.2 How is Trojan Horse Protection Achieved? 29

4. Need for Effectiveness 31

    4.1 Ineffectiveness of Conventional Systems 31
    4.2 The Security Kernel Approach 32

5. Review of Existing Approaches 35

5.1 ARPANET 35
5.1.1 ARPANET TELNET and FTP Protocols 36
5.1.2 RSEXEC 36
5.1.3 National Software Works 37
5.2 Network Security Centers 38
5.3 Military Networks 42
5.4 Dynamic Process Renaming 43
5.5 External Security Monitors 45

6. Lattice Model in a Decentralized System 49

7. Basic Message Passing Protocols 53

7.1 Protocol Design 53
7.1.1 Basic Packet Communication 53
7.1.2 Adding Security to the Basic Protocol 54
7.1.3 Trojan Horses in the Sending Host 57
7.2 One-Way Communication 59
7.2.1 Rationale 59
7.2.1.1 Military Airlift Command Example 59
7.2.1.2 Corporate Planning Example 60
7.2.2 Implementing One-Way Communications 61
7.2.3 Limitations of One-Way Communications 62
7.2.3.1 Lack of Reverse Communications 62
7.2.3.2 Protocol Difficulties 64

8. Naming under the Lattice Model 69

8.1 Reed's Generic Naming Scheme 71
8.1.1 Goals of the Naming Scheme 71
8.1.2 Basic Implementation 72
8.1.3 Garbage Collection 75
8.1.4 Other Topics 75
8.2 Incorporating the Lattice Model in the Naming Scheme 76
8.2.1 Notation 77
8.2.2 Creating Edges and Vertices 77
8.2.3 Using Directories and Services 79
8.2.4 Explicit Deletion of Edges and Vertices 81
8.2.4.1 Multics Style Naming 83
8.2.4.2 CAL Style Naming 84
8.2.4.3 Naming With Revocable Capabilities 84
8.2.4.4 UNIX Style Naming 85
8.2.5 Garbage Collection 86
8.3 Synchronization Without Writing 87

9. Downgrading Information 89

9.1 Why Downgrade Information? 90
9.2 Formularies 92
9.3 Secure Intelligent Terminals 95
9.3.1 User Requirements 95
9.3.2 Implementation 97
9.3.2.1 Processor and Memory Configuration 98
9.3.2.2 Display Windowing 99
9.3.2.3 Physical Protection 100

10. Administration of the Lattice Model 101

10.1 Proliferation of Access Classes 101
10.2 Assignment of Categories and Clearances 104

11. Limitations of End-to-End Encryption 107

11.1 The Problem 108
11.2 Countermeasures 110
11.2.1 Packet Length 111
11.2.2 Destination Address 111
11.2.3 Time Between Packets 112
11.3 Dynamic Key Renaming 113
11.3.1 Name Generation 114
11.3.2 Sychronization 115
11.3.3 Opening Connections 118

12. Authentication 119

12.1 Forwarding Authentication in the Lattice Model 119
12.2 Forwarding Authentication in Discretionary Systems 120

13. Conclusions 125

13.1 Where Have We Been? 125
13.2 Where Can We Go? 128
13.2.1 Implementation 128
13.2.2 Legislation 129
13.2.3 Further Research 130

References 131

LIST OF FIGURES

Figure 5.1 Network Security Center Failure 40
Figure 7.1 Packet Flow Through a Kernel Based Switch 56
Figure 7.2 One-Way Communications 67
Figure 8.1 Typical Naming Network 73
Figure 8.2 Deletion Example 81
Figure 11.1 Typical Packet With Key Name 114
Figure 11.2 Transmission Name Generator 116
Figure 11.3 Reception Name Generator 117

[This page intentionally left blank.]

Chapter One

Introduction

Decentralized computing systems are becoming more and more common throughout the computing industry. Although we have had decentralized systems of one sort or another since the development of the SAGE air defense system <Everett57> in the 1950's, the recent dramatic reductions in the cost of computing hardware have led to a growing feeling that decentralized computing systems offer a number of advantages in providing efficient and economical computational power to the user.

The need for protection of information in computer based systems is clear. The numerous examples of computer related crimes <Parker73>, the need to protect national defense information, and the recent passage of laws guaranteeing the protection of personal data <Privacy74> have all led to a growing awareness and concern for computer security. Decentralized systems can have an adverse impact on the security of an individual computer system by:

> "Potentially making the security controls on a specific host irrelevant by making information accessible to other hosts that do not have effective security controls," and by
>
> "Introducing additional vulnerabilities through the lack of effective security controls in network elements, e.g., insecure network communications processors." <Schell76>

However, a decentralized computing system can also enhance the security of some computational tasks. Decentralization of computing resources can introduce protection through physical separation, and a properly designed communications subsystem can ensure confinement of sensitive information within selected boundaries.

1.1 What is a Decentralized Computing System?

The term "decentralized computer system" is used in this thesis, primarily because the term "distributed computer system" has come to mean all things to all people. Distributed computing can refer to anything from a network of heterogeneous systems like the ARPANET to the IBM Attached Support Processor system in which a 360/40 handled I/O functions for a 360/65 batch processor. Therefore, to assure a more precise understanding, the term "decentralized computing" will be used throughout this thesis.

The types of systems to be considered in this thesis as decentralized systems are quite varied. They range from networks of independent heterogeneous systems, such as the ARPANET, to collections of geographically distributed processors, all performing a single special purpose function. (1) Simple time sharing systems and tightly

---

(1) SAGE <Everett57> is an example of such a dedicated single function decentralized system. Each SAGE center was capable of passing aircraft tracks to neighboring centers and correlating tracks of the same aircraft computed by two different centers. SAGE is certainly not a very interesting system from the point of view of research in the functionality of decentralized systems except for a historical

coupled multiprocessing systems are not of interest. Similarly, remote terminals with simple editing or "fill in the blanks" capabilities are not of interest. However, intelligent terminals with a significant internal processing capability are of interest. Components of decentralized systems usually can run autonomously, and often are under independent administrative control.

1.2 Why Non-Discretionary Access Controls?

The primary emphasis of this thesis is on non-discretionary access controls, access controls that are determined by the management of the computing facility and may not be changed at the discretion of the ordinary users. This emphasis on non-discretionary controls exists for two reasons. First, decentralization of the computing systems introduces new problems for a non-discretionary access control system. Different host computer systems may have different non-discretionary authorizations, yet still wish to communicate. Second, and perhaps the more important reason, formal statements can be made about the security of a non-discretionary system that cannot be made about the more general discretionary systems. Since non-discretionary access controls can effectively model a wide variety of security policies that match many real world requirements, restricting the view to non-discretionary access controls does not seem unreasonable.

---

perspective. However, from a security and protection point of view, systems such as SAGE have many of the same characteristics as more sophisticated decentralized systems.

1.3 Plan of the Thesis

This thesis examines the issues and requirements of non-discretionary access controls in decentralized computing systems, to develop a consistent approach to the protection of information. Chapters 2 and 3 outline the basic protection goals for decentralized systems and explain the rationale for the use of non-discretionary access controls. Chapter 4 describes the security kernel technology upon which much of this thesis depends, and chapter 5 summarizes much of the related work on providing security in decentralized or network-based systems. Security weaknesses in a number of these approaches are identified.

Chapter 6 outlines the basic scenario under which the lattice model will be enforced in a decentralized system. Chapter 7 discusses the basic message passing protocols under the lattice model. Chapter 8 outlines a service naming scheme that maintains consistency under the lattice model. Chapter 9 discusses the problem of downgrading information from one security level to another, and suggests how decentralized processing can aid in this task. The administrative aspects of applying the lattice model in a geographically and administratively decentralized system are examined in Chapter 10. The interactions of the lattice model and end-to-end encryption are covered in Chapter 11, and finally authentication is covered in Chapter 12.

Chapter Two

Protection Goals for Distributed Systems

Before we can examine techniques for assuring protection of information in decentralized systems, we must understand what we mean by protection. This chapter and the next present a basic set of security requirements that model much of what people mean when they say they want protection of information. Unfortunately, certain aspects of the protection problems must be excluded from consideration, because their solutions are intractable.

2.1 Basic Requirements

There are three basic requirements for information security in computing systems:

a. Information shall not be released to unauthorized individuals.

b. Information shall not be entered or modified by unauthorized individuals.

c. The services of the computing system shall not be denied to authorized individuals by unauthorized individuals.

Several interesting points should be noted about these requirements. First, the requirements are stated in a negative form. They state properties that a system must not have. Second, the requirements refer

only to individuals, that is, human beings. They make no reference to programs or processes or jobs. Third, the requirements do not define the threat environment of the system. How far are unauthorized individuals likely to go to achieve their illicit goals?

2.2 Threats

For purposes of this thesis, we shall assume a high threat environment exists for the decentralized computing system. Unauthorized, malicious individuals or organizations are assumed to exist that are willing to invest large sums of money and to commit illegal acts to obtain information illicitly. Such high threat environments exist for national defense information and for high value civilian data such as electronic funds transfer, stock transfer, or trade secret information. Malicious individuals may attempt to gain physical access to computing facilities or storage media, they may attempt to attack the communications media, or they may attempt to attack the software of a computing system.

2.2.1 Threats to Physical Security

The simplest attacks on the security of computing systems (decentralized or otherwise) are direct physical attacks. If an unauthorized individual can gain access to the front panel of a computer or can steal or copy storage media, then there is little or nothing that

can be done to protect the information. (1) Physical security is not the major topic of this thesis. Therefore, it is assumed that adequate physical protection of facilities is provided by some combination of guards, walls, fences, alarms, etc.

Other aspects of physical security that must be considered include emanations security and erase procedures. Emanations security refers to protection against electromagnetic or acoustic emanations from electronic equipment that may reveal the contents of the data being processed. Erase procedures are required for magnetic storage media that may be released or disposed of after use. It will be assumed in this thesis that adequate erase procedures are used, and that emanations security is assured throughout the decentralized computing system (including both central processing facilities and remote terminal sites). Department of Defense guidelines on emanations security and erase procedures can be found in <DoD73>.

2.2.2 Threats to Communications Media

Although physical security can be assured at the various nodes of a decentralized computing system, it is generally impossible to guard the communications links between nodes that may extend over thousands of

---

(1) One could certainly encrypt information before it is stored on easily portable (and therefore easily stealable) storage media. However, the encryption mechanism and encryption keys must be physically present somewhere in the computing system, and therefore may also be subject to theft.

miles of cables or may even include radio or satellite radio links. Not only can a hostile agent listen in to communications, but the agent can also introduce spurious messages into the communications medium.

Most often, encryption is used to protect data in a communications medium. Encryption systems normally transform data (called cleartext or plaintext) into a non-intelligible form (called ciphertext) that is then transmitted. A basic introduction to cryptography can be found in <Kahn67>.

In computer communications systems, two basic types of encryption are typically used: link encryption and end-to-end encryption. In link encryption, each individual communications link is equipped with a pair of encryption devices. Messages appear in plaintext in individual switching nodes, but are always encrypted on communications links. Link encryption is the most commonly used form of encryption today.

In end-to-end encryption, a message is encrypted before it is inserted into the communications network, and it is not decrypted until it reaches its destination. Thus, switching nodes see only the ciphertext form of messages.

The use of encryption in decentralized computing systems is discussed more thoroughly in <Kent76> and <Diffie76> and will not be covered in detail in this thesis, with the exception of chapter 11. Throughout this thesis, it will be assumed that all communications are encrypted, either with link or end-to-end encryption.

One important point must be noted here. Encryption is not a panacea. If other security controls are inadequate, then it may be possible to gain surreptitious access to cleartext information, either by attacking the system while it is processing cleartext, or by subverting the encryption mechanism itself to decrypt the material on demand.

2.2.3 Software Related Threats

The primary focus of this thesis will be on software related threats to decentralized computing systems. Software threats can be categorized into direct attacks and so-called "Trojan Horse" attacks.

2.2.3.1 Direct Attacks

Direct attacks on the software security controls of a system exploit the fact that most software systems have bugs. For example if a legitimate user of a system can find a flaw in the implementation of the security controls, then that user can exploit the flaw to gain access to information to which the user was not authorized. Some of the classes of direct attacks are described by Anderson <Anderson72>. The most interesting fact to note is that there are no published reports of a major commercial operating system withstanding a direct attack on its software security controls.

A common but often ineffective response to direct attacks on software security controls is to note that such direct attacks generally require an on-line programming capability. Therefore, it is often assumed that the system could be made safe if the users were confined to either only a restricted higher order language or only a query oriented data management system.

<Anderson72> shows the vulnerability of a system to direct attack from a restricted higher order language. Anderson successfully penetrated the Honeywell 635/GCOS III Time Sharing System from a restricted FORTRAN subset that barred the use of subroutine calls and file I/O statements. All that Anderson required to gain access to the system password file were FORTRAN arithmetic assignment statements and ASSIGNED GOTO statements.

One could claim that FORTRAN is not the proper language, and that a language such as Euclid <Lampson77>, or CLU <Liskov77>, or ALPHARD <Shaw77> could prevent direct attacks on the security controls. However, the compilers for such languages will tend to be sufficiently complex, and will change sufficiently often, that efficient and verified correct compilers cannot be expected for many years. (1)

Chapter 4 briefly outlines the security kernel technology, which is the most promising approach to countering the threat of direct software attacks. The security kernel addresses direct attacks through the use

(1) That is not to say that languages such as Euclid, CLU, or ALPHARD could not be used to produce verifiable programs now, only that the compilers will not be verified for some time.

of software that has been verified using mathematical proof of correctness techniques.

2.2.3.2 Trojan Horse Attacks

Going beyond the restricted higher order languages, the query oriented data management systems would seem to be resilient to direct attacks, if they are implemented correctly. (Most such query systems are not implemented correctly and have their own security flaws.) However, such extremely restricted systems that presumably have no accidentally introduced security flaws, fall victim to the so-called "Trojan Horse" attacks in which clandestine security flaws are deliberately introduced into the software. (1) Clandestine software modifications may be introduced at any point in a system's life cycle. They may be introduced during software development, distribution, or maintenance, by either the individuals responsible for the development, distribution, or maintenance, or by anyone who may successfully attack the computer systems used for development, distribution, or maintenance.

For example, the query system could be attacked by a "Trojan Horse" in the underlying operating system's teletype input handler. The "Trojan Horse" would scan all input from the user prior to giving the characters to the query system. If the user ever typed a particular unique pattern that served as a password, then the "Trojan Horse" would

(1) This class of attack was first identified by D. Edwards in <Anderson72>.

allow the user to access data without going through the query system. Such "Trojan Horses" are described in more detail by Karger and Schell in <Karger74>. Karger and Schell also demonstrated the ease of insertion of clandestine software modifications, by placing such a modfication in the Honeywell Multics operating system. That particular modification escaped detection during quality assurance and was distributed to all systems in the field. (1) In the next chapter, we introduce non-discretionary access controls that provide a mechanism for combatting the "Trojan Horse" threat.

2.3 Authentication

In any type of security system, the identity of the user must be authenticated prior to granting the user access to the computing system. In a decentralized system, a user wants to be authenticated on one system, and have that authentication be forwarded automatically to other systems on which the user is authorized. The technology for user authentication has been studied extensively elsewhere and will not be covered in depth in this thesis. Cotton and Meissner <Cotton75> describe a wide range of user authenticators ranging from simple passwords to magnetic stripe credit - type cards to fingerprint or voiceprint readers to genetic code readers. Richardson and Potter <Richardson73> describe in detail one authentication scheme using a combination of passwords and magnetic stripe cards.

---

(1) The clandestine modification was in fact benign, in order to avoid actual damage to customer systems.

Authentication need not even be performed by the computer. At the Air Force Data Services Center (AFDSC) in the Pentagon, terminals are segregated into rooms by their authorized security levels. A guard controls entry to each room assuring that only properly cleared users ever enter the rooms. Since the terminals in the rooms are uniquely identified to the central computer by link encryption, (1) the central computer can assume that any user on a specific terminal is cleared to the highest access class for which the terminal is authorized. In theory, no further authentication would be required. In fact, the AFDSC does also require password checks. However, the passwords serve only as a redundant check. Passwords are not the primary authenticators. The AFDSC procedures are described in <Burke74>.

Regardless of the type of authenticator chosen, authentication information must be passed from host to host in a decentralized computing system. Chapter 12 describes two schemes for forwarding authentication between host computer systems.

(1) Link encryption assures that any intruder on the communications link could not generate intelligible commands to the host. The use of encryption for authentication is described in more detail in <Kent76>.

[This page intentionally left blank.]

Chapter Three

Semantics of Access Control

Before we can propose mechanisms to enforce protection of information, we must have the semantics of the desired access control policy clearly defined. Without a clear understanding of the policy to be enforced, one has no basis on which to assume that one's protection mechanisms will serve any useful purpose. The vague security goals discussed in the previous chapter are inadequate to precisely define the requirements for a secure computing system (decentralized or otherwise).

In this chapter, we shall examine the two primary models of access control - discretionary and non-discretionary. We shall also see that, in general, formal statements of security can only be made about non-discretionary systems.

3.1 Discretionary Access Control

A very general model of access control is Lampson's access matrix <Lampson71> in which the access rights of each subject to each information containing object are defined in entries of the matrix. Normally, subjects are represented by rows of the matrix and objects by the columns. By introducing attributes such as "owner" or "control", the matrix can define not only access rights to objects, but also access rights to change entries in the access matrix itself.

Two generic implementations of the access matrix have evolved that encompass most actual computer security systems. Treating the access matrix by columns, we get an Access Control List (ACL) system such as is used in Multics <Organick72>. Each object has an associated ACL that lists the access rights of subjects. When a subject wishes to gain access to an object, the ACL must be consulted to determine access rights.

If the access matrix is treated by rows instead of columns, we get a capability system such as is described by Fabry <Fabry74>. Each subject has a capability list that describes the objects to which the subject has access rights. When a subject wishes to access an object, the subject merely invokes the appropriate capability. Possession of the capability implies that the subject has access rights to the object.

Both the ACL and the capability systems are usually implemented as discretionary access control systems. By discretionary, we mean that the "owner" of an object can determine at his or her own discretion who may have access to information containing objects. For example, in the Multics implementation of ACL's, an ACL may be modified by any user who has modify permission to the directory containing that ACL. No constraint is placed on the user as to what may be placed on the ACL. Similarly, a process that owns a capability can give that capability to any other process, again without constraint.

The basic problem of discretionary controls is, of course, their vulnerability to attack by "Trojan Horses." ACL's and capabilities must be manipulated by programs - programs that may contain "Trojan Horses." Such "Trojan Horse" laden programs could surreptitiously modify an ACL without the ever realizing what had happened. For example, the Multics PL/I compiler must change the ACL of the segment into which the object code is placed. First, the compiler must set the ACL to read-write to be able to store the new machine instructions. Then it must set the ACL to read-execute to enforce thee Multics pure procedure conventions. A "Trojan Horse" in the compiler could surreptitiously add other names to the ACL without difficulty. (1)

---

(1) The Multics compilers could easily be changed to not modify ACL's, but to get "temporary write" permission to the object segments. However, it is still true that programs subject to "Trojan Horse" attack must manipulate ACL's. The user may choose to borrow a program to modify the ACL's of all segments that match some particular selection criteria. Such a program must modify ACL's and could contain a "Trojan Horse."

The limitations of discretionary access controls have been formally modeled by Harrison, Ruzzo, and Ullman <Harrison76>. Harrison, et al. show that for a fully general access matrix, certain security questions are undecidable. In particular, they show that the so-called "confinement problem" is one such undecidable problem. The "confinement problem," as stated by Lampson in <Lampson73>, asks whether there exists a mechanism by which a subject that is authorized access to an object can leak the information contained in that object to some other subject that is not authorized access. If it can be shown that no such mechanism exists for a particular security system, then that security system is not vulnerable to "Trojan Horses." Harrison's results show that discretionary security systems may be vulnerable to "Trojan Horse" attacks, and that in general, it is impossible to determine if an information leak exists.

3.2 Non-Discretionary Access Control

Harrison, et al. point out in their paper that although the confinement problem is undecidable for a fully general access matrix, there exist a large number of security systems for which the confinement problem is decidable. Lipner <Lipner75> and Denning <Denning76> have shown that under the so-called "lattice security model," the confinement problem is decidable. The lattice model derives originally from the military classification system and is a non-discretionary access control system. Objects are assigned access classes and subjects are assigned

clearances. For a subject to gain access to an object, it must be "cleared" for the the object. A subject does not have the discretion to grant access to objects to other subject who are not cleared for the objects.

3.2.1 Definition of the Lattice Model

The fundamental basis of the lattice model is a set of partially ordered access classes from which subject clearances and object classifications are chosen. The particular interpretation of the access classes is not critical, as long as a partial ordering can be assigned. The lattice requires only that there be a lowest access class that is $\leq$ any other access class and a highest access class such that any access class is $\leq$ the highest access class. Two arbitrary access classes need not have a <, >, or = relationship, but may be disjoint.

One very simple lattice consists of two access classes - SECRET and PUBLIC. The ordering (which in this case is a total ordering) is PUBLIC < SECRET.

A more commonly used lattice is the military security lattice. In the military lattice, an access class has two components - a sensitivity level and a category set. The sensitivity levels are UNCLASSIFIED, CONFIDENTIAL, SECRET, and TOP SECRET. Categories represent compartmentalization of each sensitivity level into collections of information that require special access permission. To gain access to a

category, one must not only be cleared for the sensitivity level, but one must also be authorized the category. Examples of categories include NUCLEAR and NATO. If an object is classified SECRET-NATO, then even if a subject has a TOP SECRET clearance, the subject cannot gain access to the object unless the subject has been authorized NATO access. Since information may reside in multiple categories, an access class consists of a sensitivity level and a set of categories. Access class A is < access class B, if and only if A's sensitivity level is less than B's, and A's category set is a subset of B's. Based on this definition, only a partial ordering exists, since two access classes may be <, =, >, or disjoint. A lowest access class (UNCLASSIFIED-no categories) and a highest access class (TOP SECRET-all categories) both exist, making the system a lattice.

More complex lattices could be constructed to model other types of security systems such as corporate proprietary systems or systems subject to the Privacy Act of 1974. Turn <Turn76> describes several proposed privacy protection schemes that are based on an ordered set of sensitivity levels. One could also assign categories to each particular type of personal data. A common sensitivity level system could be used, or each category could have its own sensitivity levels, independent of other categories. As long as the partial ordering is maintained, essentially arbitrary security lattices can be defined.

For the remainder of this thesis, the lattice based on the sensitivity level and category set combination will be used. This particular lattice is commonly used (by the military) and is

representative of the basic properties of lattice models. In particular, it exhibits access classes that are disjoint, and therefore are neither less than, greater than, nor equal to each other.

3.2.2 How is Trojan Horse Protection Achieved?

Thus far, we have defined the lattice model, but we have not shown how the "Trojan Horse" threat is countered. Lattice type systems have been formally modeled by the MITRE Corp. <Bell75> and Case Western Reserve University <Walter75>. Out of these models have come two properties that must be enforced to assure invulnerability to "Trojan Horses."

First, the simple security property requires that if a subject wishes to read (or execute) an object, the access class of the object must be $\leq$ the access class of the subject. Informally, a subject must be cleared to read an object.

Second, the confinement property (1) requires that if a subject wishes to write an object, the access class of the subject must be $\leq$ the access class of the object. Thus, a "Trojan Horse" can never write information at a "lower" access class and can do no damage. The detailed motivation for the confinement property is discussed in more detail in <Bell75>.

---

(1) The confinement property was originally called the *-property by Bell and LaPadula <Bell73>.

As an alternative to the confinement property, Weissman's ADEPT-50 system <Weissman69> enforced a "high water mark" rule. Every subject in ADEPT-50 had a current access class parameter that was the maximum access class from which that subject had ever read information. The current access class moved up as the subject read higher access class material. Thus, the name "high water mark" came from the fact that the current access class could move up, but not down.

Whenever a subject S wished to read an object O, the current access class of S was set to the maximum of the access class of O and the current access class of S. Of course, if the maximum access class of S was less than the access class of O, then access would be denied. If S wished to create a new object O', the access class of O' would be set to the current access class of S. ADEPT-50, unfortunately, did not control writing into already existing objects, and so could not completely assure confinement of Trojan Horses. However, the "high water mark" system could be easily modified to include a rule that if a subject S wishes to write into an already existing object O, the access class of S must be equal to the access class of O. It should be noted that O cannot be upgraded to the access class of S. Such an upgrade would be visible to subjects at a lower access class than S, and therefore, would constitute a form of communication that could be exploited by "Trojan Horses."

Chapter Four

Need for Effectiveness

Thus far, we have examined the goals of security systems, and we have proposed mechanisms to enforce desired security policies. However, even the best security policy is worthless if its implementation is not effective and complete. In this chapter, we shall briefly summarize how computer security systems are penetrated and how effective security can be achieved. The security kernel technology, upon which most of this thesis depends, is briefly described.

4.1 Ineffectiveness of Conventional Systems

Numerous penetration studies have demonstrated that conventional computing systems do not have effective security controls. In the published literature, such systems as Honeywell GCOS <Anderson71>, IBM OS/360/370 <Abbott76>, IBM VM/370 <Attanasio76>, Bolt Beranek and Newman TENEX <Abbott76>, Univac Exec 8 <Abbott76>, and Honeywell Multics <Karger74> have been examined and found lacking in effective security controls. Further, only a small percentage of all system penetrations are reported in the literature. The evidence of successful penetration is usually kept under close guard by most managers, and thus little ever reaches the published literature.

Based on the results of the numerous and highly successful penetration studies, it can be seen that there exist fundamental security weaknesses originating from the basic complexity of conventional operating systems. Even if every known security weakness in a particular system were repaired, there would be no basis on which to believe that every weakness had been found. Further, the modifications to repair the security vulnerabilities are often so complex that they themselves may introduce new vulnerabilities. Anderson <Anderson72> reports that after extensive security "repairs" had been undertaken for one large commercial system that had been penetrated, the newly "repaired" system succumbed to a new penetration after less than one person-week of effort.

4.2 The Security Kernel Approach

To overcome the weaknesses of conventional systems, an approach based on the use of security kernels was proposed <Schell73> to assure the effectiveness of the security controls of future systems. The security kernel of an operating system mediates all accesses to information, assuring that the desired security policies are enforced.

The security kernel approach provides effective security controls by modularizing large and complex operating systems into security and non-security relevant portions. By sufficiently simplifying the security mechanisms and isolating them from the rest of the operating system in the so-called security kernel, it becomes possible to

mathematically verify the correctness of the security mechanisms. To assure completeness, the design of the security kernel must be based on a formal model of secure systems <Bell75>. The verification methodology takes the kernel design from the formal model to actual binary machine code in several steps, with correspondence proofs between each intermediate representation. The verification methodology is discussed in more detail in <Millen76>.

Security kernels have been implemented for the PDP-11/45 by the MITRE Corp. <Schiller75> and by U.C.L.A. <Popek74>. Kernels are presently under development for the UNIX operating system <Ritchie74> by both the MITRE Corp. <Biba77> and U.C.L.A. <Kampe77>. Kernels were under development for the Honeywell Multics system <Schroeder75> and for the Honeywell Series 60 Level 6 minicomputer <Honeywell76>, but Headquarters, United States Air Force Systems Command has directed that these two efforts be terminated in 1977, prior to completion. However, the United States Air Force SATIN IV packet switched network is using portions of the security kernel technology in the Internal Access Control Mechanism (IACM) present in each SATIN IV communications processor.

The security kernel technology forms the essential basis for much of this thesis. To date, the security kernel is the only approach identified to provide effective security controls for large complex systems. However, since most existing systems do not have security kernels, we shall examine configurations of decentralized systems in which strategic placement of security kernel based communications

processors can provide effective security controls to systems without effective internal access controls.

Chapter Five

Review of Existing Approaches

This chapter is a review of several existing or proposed approaches to security in decentralized computing systems. Problems and drawbacks of several of the approaches are identified. Later chapters will address these problems and propose some solutions. Because some of the problems are inherently unsolvable, solutions will not be proposed for all the problems.

5.1 ARPANET

In this section, we examine four protocols for implementing security in the ARPANET - TELNET, FTP, RSEXEC, and the National Software Works (NSW). Of course the ARPANET Interface Message Processors (IMP's) were not developed to be "penetration-proof." While IMP software is quite reliable, it has not undergone the formal verification necessary to assure security. In addition, IMP-IMP communications lines are not encrypted. Therefore, the ARPANET is presently vulnerable to wire tapping. ARPA is presently sponsoring development of two end-to-end encryption devices for the ARPANET. One is called the Private Line Interface (PLI) <IMP76>, and the other is called the BCR (Black - Crypto - Red) <Bressler76>.

5.1.1 ARPANET TELNET and FTP Protocols

The existing ARPANET TELNET and FTP protocols <Feinler76> (1) provide very limited support for protection of information. Each host is responsible for assuring its own protection, primarily by requiring password authentication at the time a network connection is made. Using these protocols, a user must remember different passwords (2) for each machine used and must transmit these passwords through various host machines, leaving opportunities for the passwords to be stolen. Alternatively, the user could store passwords for the foreign machines in files on the local machines. This technique, however, extends the vulnerability of the passwords.

5.1.2 RSEXEC

RSEXEC (the Resource Sharing Executive for the ARPANET <Thomas73>) provides a much more sophisticated decentralized environment than the TELNET and FTP protocols do. RSEXEC allows a user to view the file systems of several machines on the ARPANET as a single file system. (3)

---

(1) TELNET is a protocol to provide remote terminal communication over the ARPANET. Using TELNET, a terminal connected either to a host processor or a terminal concentrator can communicate with any host on the network. FTP is a protocol to provide file transfer capabilities between hosts on the ARPANET.

(2) Users often choose the same password for all sites, making it possible to attack several sites, after stealing a password for only one site.

(3) Currently the RSEXEC protocols are only completely supported for the

The user can name files at distant sites and local files uniformly. However, from an authentication point of view, RSEXEC is very similar to TELNET and FTP. Passwords must be stored on user machines and transmitted to server machines whenever network connections are made. Although RSEXEC hides much of the password processing from the user, the stored passwords for other systems remain subject to attack, either in the user system or while being transmitted through the network.

5.1.3 National Software Works

The National Software Works (NSW) <Millstein76> is another decentralized computing facility being implemented on the ARPANET. The NSW is intended to support software development activities by providing access to software development tools resident on various hosts. As in RSEXEC, the user of the NSW is sheltered from the issues of where on the network particular files or tools are stored. Authentication is very simple in the NSW. A user who wishes to login to the NSW first connects to a local Front End (FE) process running on a local machine. The FE delivers the user's login request and authentication password to the Works Manager (WM), which runs on some centralized machine in the NSW. All requests for service must go through the FE to the WM where the user's identity is verified. The WM then forwards authorized service requests to appropriate systems. All systems must trust the WM

TENEX operating system. RSEXEC is partially supported by the ITS and Multics operating systems.

implicitly. No authentication is performed by the target host, but rather any WM request must be honored without question. A host can assure itself that it is talking to the real WM by a scheme of dedicated network sockets, but all systems must accept the trustworthiness of the central WM.

5.2 Network Security Centers

One approach to security in decentralized systems is the concept of a Network Security Center (NSC) proposed by Branstad <Branstad73, Branstad75> and expanded upon by Heinrich and Kaufman <Heinrich76>. The NSC is a centralized facility that, as proposed by Branstad, provides a secure cryptographic key distribution service. As such, the NSC can assure identification and authentication of the user to the servicing host computer and vice versa. The authentication is implicit in the cryptographic keys. The NSC is an example of a specialized multilevel host. It does not support online programming, but must be verified to be free of "Trojan Horses."

Note that the function of the NSC is analogous to the function of the Works Manager (WM) in the National Software Works (NSW). While such a centralized authority may be acceptable to a network under a single management control, it may not be at all acceptable in a decentralized computing system that does not have central management. Diffie and Hellman <Diffie76> suggest an approach to avoid the necessity of trusting a single central authority. They suggest the use of multiple

independent NSC's, each of which verifies and authenticates a requested two-way communication. Each NSC provides a cryptographic key to the sender and receiver processes. All the cryptographic keys are combined via addition modulo two to produce a key not known to any of the NSC's involved. Thus, the sender and receiver processes need not trust any single NSC not to release their cryptographic key. Only if all the NSC's cooperate, can they compromise the cryptographic key that was generated by the processes.

Heinrich and Kaufman, however, attribute the NSC with security capabilities beyond those proposed by Branstad. In particular, they claim that the NSC can prevent unauthorized access to data by legitimate users of the network. However, the NSC is in fact unable to prevent such unauthorized accesses, unless the various host processors are themselves secure. The following two examples demonstrate the inability of the NSC to prevent unauthorized access.

For the first example, assume a network consisting of three host computers - A, B, and C. (See figure 5.1.) Assume A, B, and C do not have effective internal security controls, but that they communicate only via the network using cryptographic keys provided by the NSC. Assume A is authorized to access B's data base, and B is authorized to access C's data base, but A is not authorized to access C's data base. The NSC enforces the protection of C's data base by not providing a common cryptographic key to A and C. However, the NSC can do nothing if B forwards data from C to A.

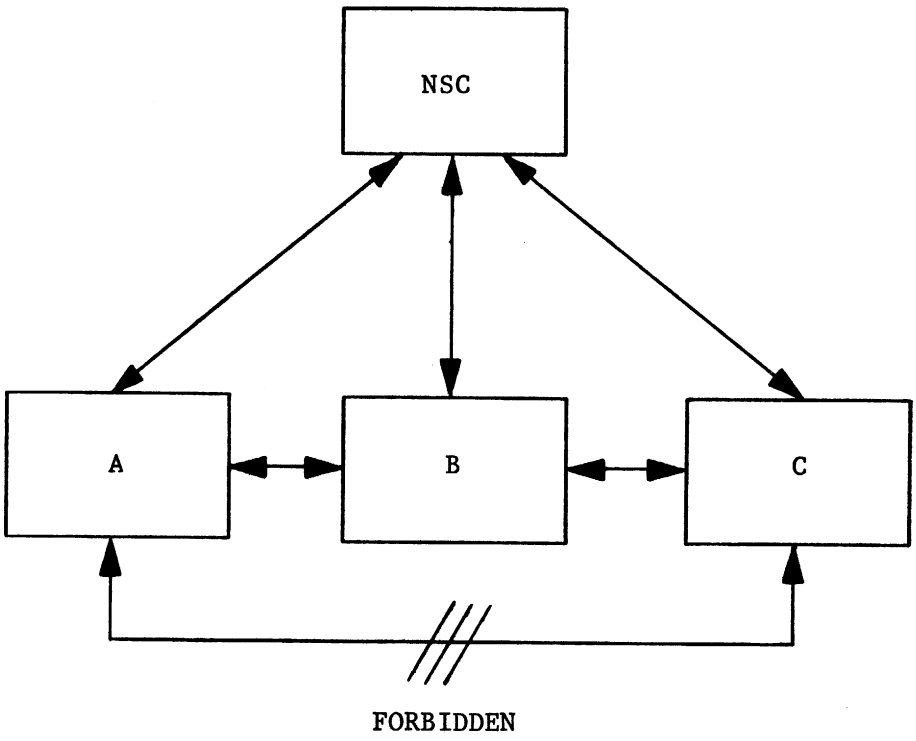


Figure 5.1 Network Security Center Failure

If host B enforced the lattice model with a security kernel, then B could implement two untrusted processes, one to communicate with A, and one to communicate with C. Since A and C may not communicate (presumably because their access classes are disjoint), the security kernel in B would prevent the two processes from communicating, and therefore prevent A from receiving information from C. However, this type of protection that the security kernel of B could provide is entirely independent of the presence or absence of an NSC.

The example above uses multiple systems connected to a network to leak information to unauthorized users. In fact, the NSC cannot prevent unauthorized access, even if there is only one system involved. Effectively, the NSC's granularity of protection is an entire computer system. If a user can gain access to a host system for some legitimate purpose, and the host's security controls are ineffective, then that user can gain access to any information in that host. Since such an attack would take place entirely within a single host, it would be invisible to the NSC.

Heinrich and Kaufman describe the NSC as maintaining an access matrix similar to Lampson's <Lampson71>. However, as shown by Harrison, et al. <Harrison76>, merely maintaining an access matrix does not guarantee that unauthorized access does not occur. In the first example, using Harrison's terminology, B can leak to A the generic right to read C's data base. In the second example, one can model the ineffective security controls of the operating system as a special subject in the access matrix. The special subject has access to all data in that particular system, but due to the ineffective security controls, all other subjects on that system have access to the special subject. Thus, if the NSC grants a user access to that particular system, by transitive closure on the access matrix, the user has been granted access to all data stored in the system. While Heinrich and Kaufman imply that the host system might choose to add additional protection, they do not make clear that unless the host itself maintains effective security controls, the NSC can leave significant security

vulnerabilities unblocked. This issue is of paramount importance if one wishes to build secure decentralized systems in which some of the component host systems are fundamentally incapable of providing effective internal access controls.

5.3 Military Networks

The U.S. Department of Defense is presently developing two packet switched networks - SATIN IV for the U.S. Air Force Strategic Air Command and AUTODIN II for joint service communications. These networks achieve protection of classified message traffic by encrypting communications on a link by link basis and providing effective security controls in each message processor. For example in SATIN IV, each message is labelled with a sensitivity level and category set and the Internal Access Control Mechanism (IACM) of each communications processor assures that messages are routed only to destinations that are properly cleared to receive them.

When messages are entered into SATIN IV from an external interfaced system, the IACM must differentiate between interfaced systems with effective security controls and interfaced systems without effective security controls. Systems without effective controls cannot be trusted to properly label messages. Messages from such untrusted systems must be treated by SATIN IV as classified at the highest access class

processed by that particular system. (1) The IACM's of each communications processor will be verified to operate correctly and provide effective security controls.

The security characteristics of AUTODIN II are less well defined at this time, but are expected to be similar to SATIN IV. A brief summary of the requirements of both the SATIN IV and AUTODIN II systems can be found in <Chandersekaran76>. One proposal for AUTODIN II security can be found in <Postel76>.

5.4 Dynamic Process Renaming

Farber and Larsen <Farber75> suggest an approach to security in a ring network by dynamically renaming the process names that appear in the message destination fields. They reason that if in a series of messages sent from one process on a host to another process on a different host, the destination fields of the messages are changed in every message based on presumably secret transformations known only to the source and destination systems, then an intruder could not follow the rapid exchange of messages and would be unable to extract information. Farber and Larsen describe synchronization and error

---

(1) When presented with a message labelled at a lower access class than the highest access class processed by the untrusted message source, the IACM must either generate a security alarm or relabel the message at the highest access class processed by that system. Operational requirements will determine which is appropriate. If relabelling is performed, the new label would consist of the alleged access class of the message and the access class to which the packet must be protected.

detection methods that make dynamic process renaming a practical communications protocol. Unfortunately, they do not address the possibility of computer assisted traffic analysis that could easily distinguish patterns in the traffic and determine the content of the transmissions. For example, the login dialog to most time sharing systems is very stylized in which the system sends a greeting message, the user responds with a login command, the system requests the user's password, and the user types it in. Such a dialog could easily be recognized in a recording containing many unrelated messages. Such an analysis was done in one penetration of a computer system, documented in <Computerworld75>, in which the penetrator examined teletype communications buffers to collect passwords of many users. An even easier example would be traffic generated by a program like the MACSYMA system <MACSYMA75> in which messages from the program to the user are sequentially numbered so that the user can reference them easily in later requests. Clearly, dynamic process renaming is effective only against very unsophisticated attacks. Encryption of communications is much more effective against sophisticated penetration attempts.

5.5 External Security Monitors

Painter <Painter75> proposes an approach to security in computer networks in which an external minicomputer is attached to each host processor to monitor all hardware and software operations for security malfunctions. To monitor the hardware, Painter proposes equipment analogous to Automatic Test Equipment (ATE) be attached to run periodic tests on all hardware components. Since hardware can fail randomly, such tests are important for the operation of any secure computer facility. Software versions of hardware security monitors have been implemented for the Honeywell 645 <Karger74> and for the Honeywell 6180 <Hennigan76>. Painter points out two major difficulties with his external hardware monitor proposal. First, ATE normally interferes with hardware performing its normal operational functions. Therefore, either ATE must be designed that does not interfere, or redundant hardware must be provided to allow checking of components on an offline basis. In the latter case, software must also exist to allow reconfiguration of the hardware without disruption of ongoing processing. Second, as LSI technology advances, it becomes more and more difficult to build ATE. Because of the concentration of functions on single chips, it becomes impossible to break systems down into separate "black boxes" for isolated testing. Perhaps future LSI hardware can be designed with additional leads for ATE interfaces. Painter also points out that his technique cannot detect Trojan Horses that may be concealed in LSI chip designs.

Unfortunately, Painter's scheme for external software monitors is less well founded than his hardware monitor scheme, because software, unlike hardware, does not fail randomly. Software security systems are either correct or incorrect from the start. They do not "fail" after a period of time. Painter proposes that software security surveillance be carried out by hardware performance evaluation monitors that examine the contents of registers and main memory "looking" for security penetrations. Painter admits the hopelessness of analyzing every operation performed by the host computer. The CPU time required would be many times that required by the host computation itself. Painter instead suggests a statistical approach, periodically checking the host for software security penetrations. However, the types of penetrations described in <Karger74> can be consummated in a matter of microseconds. The probability of statistically discovering a well rehearsed penetration is extremely small. More importantly, Painter offers no evidence that such an external software security monitor can be effectively implemented, even given unlimited CPU time. Essentially, Painter expects the monitor to examine arbitrary programs running in the host system to see if they ever enter an insecure state. One can draw an analogy to automata theory that shows the undecidability of the question of whether an arbitrary Turing Machine ever enters a particular state <Hennie77>. While a proof that Painter's approach is not effectively computable is beyond the scope of this thesis, the feasibility of his approach is certainly open to question.

Painter takes his software surveillance monitor concept one step further and suggests that the monitor could be implemented on the host system itself. This technique of self-monitoring is shown to be insecure in <Karger74>. If the host system is secure, then the self-monitor could be useful in detecting some, but not necessarily all, unsuccessful penetration attempts. However, if the host system is not secure, then the successful penetrator will immediately modify the self-monitor to assure that it only reports that all is well.

[This page intentionally left blank.]

Chapter Six

Lattice Model in a Decentralized System

Before we can study the application of the lattice security model to decentralized computing systems, we must define the characteristics of the subject system. We assume the decentralized computing system consists of a large number of host computers, ranging in size from very large general purpose systems to individual microprocessor based "personal" computers. No assumption is made concerning homogeneity of instruction sets. The host computers are interconnected using a variety of communications media including direct digital data links, store and forward message processors, broadcast links, etc. The host computers are not managed by a central authority. However, although administration of the host computers is decentralized, there is a common security lattice that is to be enforced on all machines. (1) We must assume that a very large number of security access classes will be in use in the system, although most hosts will only use a small subset of them. The large number of access classes comes from the desire to support a commercial decentralized system with thousands of customers in which each customer may wish to define several categories of information to be protected. In this context, a customer might be an entire corporation or division of a corporation.

---

(1) The apparent dichotomy between decentralized control and a common security lattice need not exist. The various divisions of a corporation may operate with a high degree of autonomy, yet all agree on a common system for protecting company confidential material. Similarly, a common system for protecting classified information exists among the otherwise autonomous agencies of the Department of Defense.

Not all host systems will be authorized to receive information at all access classes. Even if Chrysler's computer had a security kernel, Ford would be unwilling to store its data there. Therefore, the communication network must assure that information is never made available to hosts that are not authorized to receive the information.

Not all host systems will have effective security controls, because many hosts will run conventional insecure operating systems. Despite insecure software, such conventional systems can adequately protect sensitive information if they are run in a dedicated mode. The Department of Defense defines dedicated mode as follows:

> "An ADP [Automatic Data Processing] System is operating in a dedicated mode when the Central Computer Facility and all of its connected peripheral devices and remote terminals are exclusively used and controlled by specified users ... for processing of a particular type ... of classified material." <DoD73>

If a host system running in a dedicated mode is connected to a decentralized computing system, then the communications network must assure that all output from the dedicated host is treated at the dedicated access class. The dedicated host cannot be trusted to correctly mark the access class of its output.

Systems that have effective security controls are called multilevel secure systems. The Department of Defense defines multilevel security mode as:

> "A mode of operation ... which provides a capability permitting various levels and categories or compartments of material to be concurrently stored and processed ... from various controlled terminals by personnel having different security clearances and access approvals." <DoD73>

A multilevel system can effectively control access to a number of distinct access classes. However, as mentioned above, even though the multilevel system has effective software controls, it may not be authorized to receive all access classes, because the system may be under the physical control of persons not authorized certain access classes.

A very important requirement of this scenario is that the implementation of the lattice model not interfere with the basic goals of decentralized computing systems - the ability to share information and computing resources and the ability to achieve robustness by taking advantage of redundancy. Therefore, the lattice implementation should not preclude such nominally secure operations as a SECRET process reading information from an UNCLASSIFIED data base on another host system. However, the implementation must assure that a "Trojan Horse" in the SECRET process cannot downgrade information while reading the UNCLASSIFIED data base.

One final assumption must be made in this scenario. All external communications are encrypted using either link encryption or end-to-end encryption. Chapter 11 discusses encryption in more detail.

[This page intentionally left blank.]

Chapter Seven

Basic Message Passing Protocols

In this chapter we shall examine the basic message passing protocols for decentralized computing systems and show their relationships to the lattice security model. As part of the discussion, we will show how the lattice model controls can be added without adversely affecting the reliability or performance of the basic protocols. Performance of the basic message passing protocols is critical to the decentralized system, because all other protocols are built from the basic protocols in a layered fashion.

7.1 Protocol Design

7.1.1 Basic Packet Communication

Before we can discuss the basic protocol design, we must define some terminology. The basic unit of communication is the packet. Packets may be sent to and from ports that exist as logical full duplex channels on the various hosts attached to the network. A host may have a large number of ports that are simultaneously involved in network communication. The notion of a port here is taken from Cerf and Kahn's Transmission Control Program (TCP) <Cerf74>. The communications network

will deliver packets to their destination ports with high reliability, but delivery is not 100% guaranteed. Some type of retransmission strategy will be required for lost or damaged packets. This very low level protocol is similar to the datagram protocols described in <Pouzin76>. Higher level protocols can be constructed to break messages up into fixed size packets, reassemble the packets, provide for retransmission and flow control, etc.

7.1.2 Adding Security to the Basic Protocol

Adding lattice security controls to the very simple protocol described above is quite straightforward. Initially, let us assume that link encryption is used on all communications lines that must transmit sensitive information. Use of end-to-end encryption is discussed in Chapter 11. If each host attaches a security label to each packet as it is entered into the network, and if the network communications processors are multilevel secure, then the network communications processors can assure that packets are delivered to only those hosts that are authorized to read information at the access class of the packet.

Since packets pass through the communications processors in cleartext, the software in the communications processors must be prevented from maliciously downgrading information. For example, a "Trojan Horse" in the message routing software could copy a TOP SECRET packet into a newly fabricated UNCLASSIFIED packet and send the

UNCLASSIFIED packet to an uncleared host system. Such a "Trojan Horse" threat could be avoided by verifying the correctness of all software in the communications processor. This approach is taken in the AUTODIN I system. (1) Unfortunately, very severe restrictions are imposed on software development by the AUTODIN I security approach. All programmers must be cleared to the highest access class of information to be passed. No off-the-shelf software can be used. Software updates must undergo nearly exhaustive testing. Only because the AUTODIN I software is very simple and rarely changes can such restrictions be implemented. They are certainly unreasonable for more complex communications systems in which off-the-shelf operating systems and compilers are desired and in which software changes may occur frequently.

To reduce the unreasonably strict AUTODIN I security restrictions, the communications processors could be implemented with security kernels. Figure 7.1 shows a very simplified view of packet flow through a kernel based packet switching processor. Input packets are received by a trusted process that must identify the packet access class and store the packet in a memory segment of the proper access class. The input process does no more than identify the packet's access class and send a wakeup to an untrusted process (that could contain a "Trojan Horse"). The untrusted process (or processes) then performs all necessary validation, routing, and other functions normally performed by

(1) AUTODIN I is a message switching system presently operated by the Defense Communications Agency (DCA). Its security characteristics are briefly summarized in <Lipner72> and <Anderson72>.

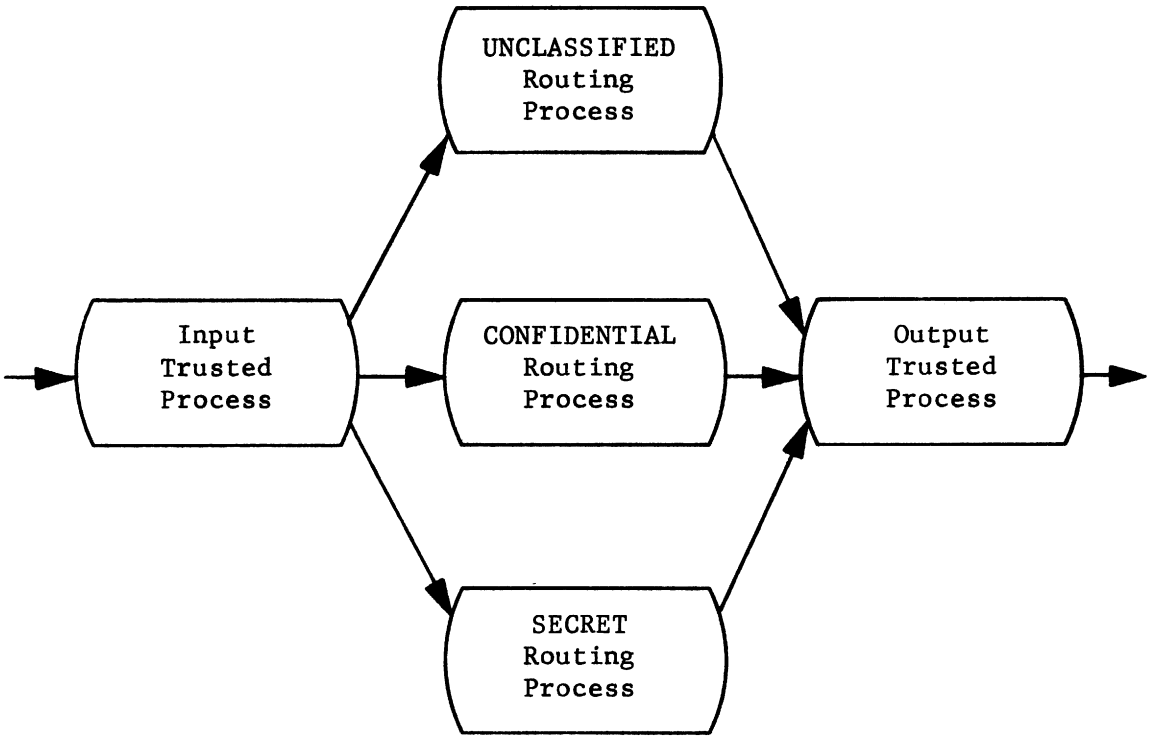


Figure 7.1 Packet Flow Through a Kernel Based Switch

a packet switch. The untrusted processes are constrained by the security kernel to obey the confinement property, and are therefore unable to downgrade information. Just before transmission, the untrusted process gives the output packet to a trusted process that verifies the correctness of the security label (since a "Trojan Horse"

could attempt to mislabel a packet) and then sends the packet out the communications line. Since communications processors may not be authorized the same access classes, the trusted process must also verify that the next communications processor intended to receive the packet is authorized to receive the packet. If the packet is being sent to a host processor, the trusted process must verify that the host processor is authorized to receive the packet.

<ESD74> contains a more detailed comparison of the security kernel approach and the AUTODIN I approach to secure communications processors. The basic issue, however, is that the communications processors can assure that packets are delivered only to hosts that are cleared to receive them.

7.1.3 Trojan Horses in the Sending Host

The protocol described in the previous section is adequate to assure security if packets are properly labeled when they enter the communications network. If the originating host processor has adequate security controls (presumably because it runs a security kernel), then packets will be correctly labeled. Just as in the multilevel packet switch, the multilevel host must have a trusted process verifying the accuracy of security labels on outgoing packets.

However, if the originating host processor does not have adequate security controls and therefore is presumably running in a dedicated mode as defined in chapter 6, then a "Trojan Horse" in the originating host could easily mislabel packets to send information classified at a high access class in a lower access class packet.

Therefore, to achieve confinement of the "Trojan Horse" in the originating host and to maintain consistency with the definition of dedicated mode, the security protocol must be modified. The input trusted process of the packet switch must know which hosts can effectively protect information and which cannot. The input trusted process must assure that all packets received from a host that runs in a dedicated mode are labeled at the dedicated access class. (1) If the input trusted process receives a mislabeled packet, it can either relabel the packet, or it can report an attempted security violation. The particular choice is application dependent and does not impact the basic security of the protocol.

---

(1) In fact, the Department of Defense also defines host systems that may operate in a controlled environment over a limited range of access classes. For example, the Air Force Data Services Center (AFDSC) in the Pentagon runs a modified version of the Multics system in a two-level SECRET/TOP SECRET controlled environment. The AFDSC Multics system (described in <Whitmore73>) is trusted to separate SECRET and TOP SECRET, but because it does not have a security kernel, it is not trusted to separate UNCLASSIFIED from SECRET or TOP SECRET. For such controlled environment systems, the input trusted process must maintain a minimum access class M, and assure that for all packets transmitted by the host labelled access class P, M must be $\leq$ P.

7.2 One-Way Communication

7.2.1 Rationale

Even though some of the hosts in a decentralized computing system do not have effective security controls and therefore must run in a dedicated mode of operation, it is still desirable to have communications with these hosts. Obviously, two dedicated hosts at the same access class can communicate freely with each other. Similarly, a dedicated host can communicate freely with an untrusted process running on a multilevel host, if the untrusted process is at the same access class as the dedicated host. However, it would be very desirable if two dedicated hosts at <u>different</u> access classes could have some limited form of communications. In particular, if there are two hosts A and B, such that the dedicated access class of A is $\leq$ to the dedicated access class of B, then it would be desirable if programs on B could gain access to the data stored in A on a read-only basis. The following two examples demonstrate the utility of such read-only communication.

7.2.1.1 Military Airlift Command Example

The US Air Force Military Airlift Command (MAC) runs three Honeywell 6080 GCOS systems as part of the World Wide Military Command and Control System (WWMCCS). Because GCOS does not have adequate

security controls, the systems must run in a dedicated mode in which two of the systems process only unclassified information and the third system processes classified. During crisis situations, certain portions of the normally unclassified data base become classified. When a crisis occurs, MAC must copy the entire data base onto a set of disk packs to be transferred to the classified machine. This manual transfer procedure is extremely time consuming and does not meet MAC's requirements for rapid response to crisis situations. In addition, unclassified updates continue to come into the unclassified system, rapidly making the classified data base obsolete. Since the purpose of the classified data base is to allow contingency planning during crisis situations, it is important to keep the data base accurate. The basic requirement is that updates to the unclassified data base be accurately reflected in the classified data base in a timely fashion. (1)

7.2.1.2 Corporate Planning Example

A similar example to the Military Airlift Command system can be imagined for a civilian corporation. Assume there exists a corporate data base system implementing some basic functions of operational control of the company. As this data base only reflects routine day to day operations, it is treated at a low sensitivity level. On a separate computer, the corporate planning staff has a data base system that is

---

(1) The description of MAC requirements was taken from Appendix VII of <Schacht76>. The original suggestion for read-only access to the data base was made by S. Lipner in <Lipner71>.

considered highly sensitive. The corporate planning data base requires periodic updates from the day to day data base, but no information is required to flow from the corporate planning data base to the day to day data base.

7.2.2 Implementing One-Way Communications

Solving the two problems posed above is relatively simple in a single multilevel computer. A process is created at the high access class, and it is granted read-only access to the low access class data base. However, in a decentralized computing system, the high access class process must send a message to a low access class process on the low access class machine in order to read the data base. What was a read-only transaction on a single machine has been transformed into a read-write transaction by the use of a communications network. (1)

In the decentralized system, the read request from the high access class host to the low access class host can be eliminated if the low access class host is preprogrammed to automatically send data base updates as they occur to the high access class host. Since the updates are sent automatically, no information need flow from the high access class host to the low access class host. In addition, the high access class host need not "poll" the data base. It receives updates only as they occur.

---

(1) In fact, the transaction was read-write on the multilevel machine as well, but the security kernel and the supporting descriptor based hardware made it possible for the read operation to occur "invisibly" to the lower access class.

Of course, this update approach works only for preprogrammed periodic transactions and offers no selectivity to the high access class host. Fortunately, such preprogrammed transaction oriented systems are common in the data processing industry. As another example, consider a radar air traffic control system. There are several computers located at radar sites connected to a central air traffic control computer via communications links. The radar site computers do nothing but convert raw radar returns into correlated air tracks that are then sent to the central facility. Assuming each radar has a fixed area of coverage, all communications flow is in one direction.

7.2.3 Limitations of One-Way Communications

Obviously, the one-way communications scheme outlined above has many limitations. These limitations fall into two basic classes discussed below - lack of reverse communications and protocol difficulties.

7.2.3.1 Lack of Reverse Communications

The primary limitation of one-way communications is inherent in its name. The fact that no reverse communications are allowed eliminates from consideration any applications that require a large amount of two-way communications. In particular, the high access class host cannot selectively query the low access class data base, based on high

access class computations. It must swallow the entire data base as it is generated and updated.

Based on these limitations, one would ordinarily dismiss one-way communications from any further consideration. However, we must remember that the ostensible purpose for considering one-way communications was to allow limited communications between inherently insecure host computers (or untrusted processes on secure hosts). While it would be desirable to use multilevel secure hosts for all applications, such hosts are not available on a commercial basis yet. Development of security kernels for large scale general purpose systems is a current research effort described in <Rhode77>. Even if multilevel secure hosts were available now, large investments in conventional insecure hardware and software would preclude immediate conversion. On the other hand, multilevel secure communications processors will be available soon outside the research environment. The Air Force's SATIN IV network is one such system. Therefore, making secure decentralized computing available on a limited basis using one-way communications seems to meet at least some interim needs. In addition, Chapter 9 will discuss some approaches to provide limited reverse communications to alleviate some these severe restrictions.

7.2.3.2 Protocol Difficulties

In addition to the limitations inherent in the concept of one-way communications, there are even difficulties in implementing the preprogrammed transaction updates. Difficulties arise in the areas of error control and flow control, although we shall see that these difficulties can be overcome.

Error detection and correction is needed, because the communications medium is not in general error free. Two generic types of error control strategies are most commonly used - feedback error control and forward error control.

Feedback error control is most commonly used in systems with a usually low probability of error, but with occasional bursts of high error probabilities. In feedback error control, the receiver examines each arriving packet for possible errors. If no errors are found, an acknowledgment is returned to the sender. If the packet is found to be in error, a negative acknowledgment is returned. The sender retransmits the packet upon receipt of a negative acknowledgment, or if no acknowledgment is received for some period of time. However, if the destination is on a dedicated host at a higher access class, then not even packet acknowledgment can be permitted, because a "Trojan Horse" could easily encode information in patterns of acknowledgments. For example, the "Trojan Horse" could generate extra acknowledgments to non-existent packets that would then be interpreted by software on the low access class system.

Forward error control is typically used in systems that have fixed error probabilities. Sufficient redundant information is transmitted with every packet to allow the receiver to reconstruct information that may have been lost or garbled. By comparison, feedback error control transmits only enough redundancy to detect errors, but not enough to correct them. Since no feedback is required, forward error control is not vulnerable to "Trojan Horse" attacks.

Flow control presents problems similar to feedback error control. The receiving host must somehow tell the sending host how fast it can receive packets. Otherwise, the sending host may transmit so fast that the receiving host's buffers overflow. A variety of flow control techniques are possible. For example, the ARPANET Host to Host Protocol <Feinler76> requires the sender to preallocate buffer space in the receiver. The sender may not transmit until the receiver confirms that the requested buffer space is available. In the Transmission Control Program (TCP) <Postel76>, the receiver tells the sender how much may be transmitted. When that so-called transmission window is exceeded, the sender must wait until the receiver extends the window. Other schemes are also possible. However, inherent in any such flow control scheme is an information flow from the receiving host back to the sending host. A "Trojan Horse" in the receiving host could exploit such an information channel to release information.

Forward error control has the major drawback that it requires significantly more bandwidth than feedback error control. The extra bandwidth is required to transmit enough redundant information to handle

the worst case error probabilities on a noisy channel. (1) Therefore, the following scheme is suggested to allow use of feedback error control and to provide flow control in a one-way communications channel without the risk of a "Trojan Horse" communicating in the reverse direction. Since it is the dedicated mode host processor that is untrustworthy, a multilevel secure front end processor with a security kernel could be inserted between the communications network and the dedicated host. (See figure 7.2.) The trustworthy front end processor could then perform feedback error control and flow control functions without fear of "Trojan Horses." The front end processor must guarantee to store and accurately deliver all packets to the host, since end to end error control is no longer possible. Therefore, the front end processor must maintain a reasonable quantity of auxiliary memory on which to store packets until the host processor accepts them.

A one bit communications channel for a "Trojan Horse" exists even with the trustworthy front end processor, because there can only be a finite quantity of memory in the front end processor to buffer packets. (2) However in practice, the front end processor running out of memory is a highly visible event, and any attempt to communicate using this channel could be very quickly detected. Therefore, the secure front end processor does seem to be a viable approach to providing error detection

---

(1) Forward error control is best used in high error probability environments in which feedback error control would breakdown in an avalanche of negative acknowledgments and retransmissions.

(2) This problem is similar to the disk allocation problem described in <Schiller75>.

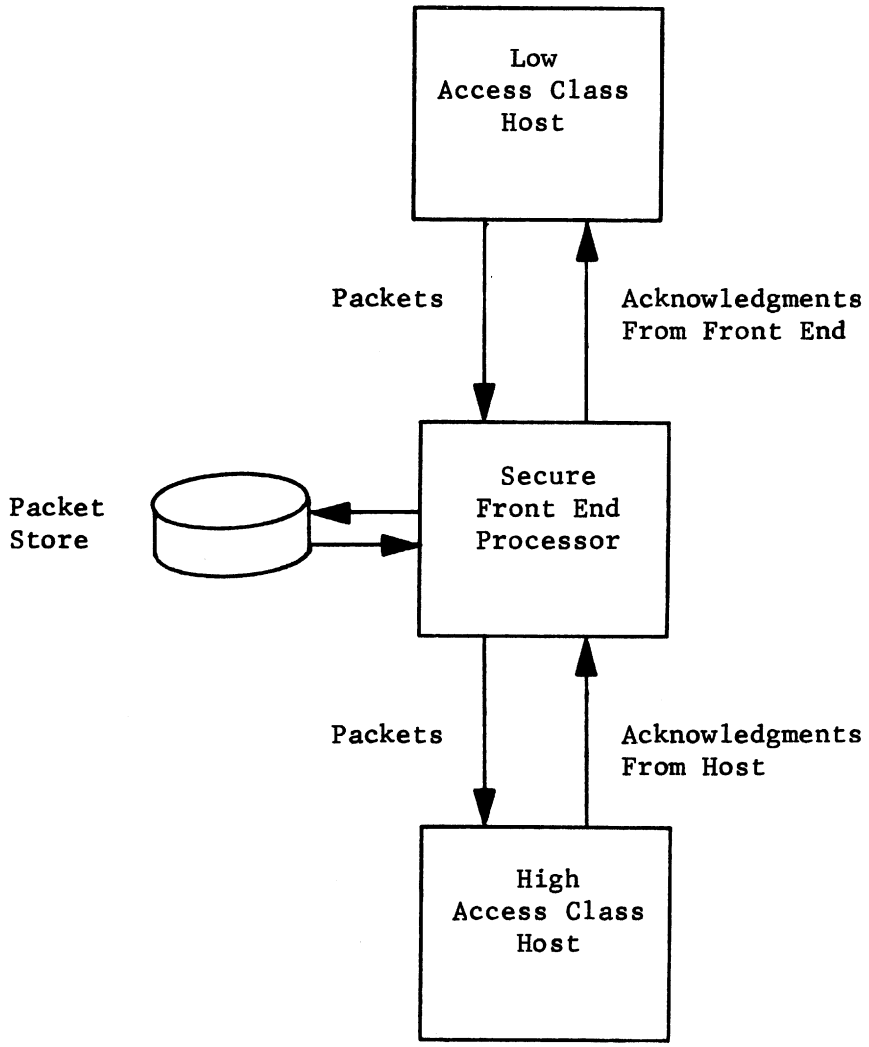


Figure 7.2 One-Way Communications

and correction and flow control securely in one-way communications. It should be noted that the packet buffering for feedback error control and flow control could be performed by the destination network switching processor, rather than by a separate front end processor. For example, the SATIN IV communications processors journal all message traffic for

possible retrieval at a later time. Such a journal could be used as the buffer store, if it had sufficiently high reliability.

Chapter Eight

Naming under the Lattice Model

One of the major limitations of networks like the ARPANET has been the lack of a uniform network wide naming scheme. Users of one host system must learn the naming conventions of any other dissimilar host systems that they wish to use. In addition, users must know the name of the host on which a service exists in order to use it. Therefore, users must be explicitly aware of redundancies across host boundaries that have been created for reliability purposes. Ideally, a user should be able to give the name of something without having to know on which host it is implemented. If multiple copies exist for reliability purposes, the decentralized computing system should automatically select one for the user.

Security of information is very closely tied to the naming structure of a system. In the Multics system <Organick72>, this tie is very explicit. The Access Control List of an object can be modified by anyone who has modify permission on the parent of the object. However, even if one separates the naming structure from the authority hierarchy as is suggested in <Rotenberg74>, naming and security must remain closely tied, because both the names themselves and the shape of the naming structure contain user modifiable information - information that is also subject to modification by "Trojan Horses".

The tie between security and naming can be seen in the evolution of the formal representations of the lattice model. In the initial MITRE security model <Bell73>, naming of objects was not considered. An attempt to implement this simple model on the PDP-11/45 <Schiller73> failed, because a "Trojan Horse" could easily communicate information in the names of objects or in the shape of the directory tree. The Case security model <Walter74> provided the first solution to this problem by recognizing that names and groupings of names, i.e., directories, must also be assigned access classes. Ames extended the Case model in <Ames74> to include all the attributes of a Multics segment, not just names. At the same time, Bell revised the MITRE model in <Bell74> to include the notion of the Multics hierarchy as an explicit element of the model.

In this chapter, we will take a broader view of naming than just the Multics directory hierarchy. In particular, we wish to include naming structures in which an object has more than one parent. We shall see that relaxing the single parent restriction of Multics generates certain constraints on the underlying host implementations of a network wide naming scheme.

8.1 Reed's Generic Naming Scheme

The naming scheme to be considered in this chapter is the one suggested by D. Reed for the proposed M.I.T. Laboratory for Computer Science Local Network. Because a description of Reed's scheme has not yet been published, this section will highlight its major aspects.

8.1.1 Goals of the Naming Scheme

Reed identifies four major goals to be achieved by a network naming system. First, the naming scheme should support translation of generic service names into specific service instances. The user should not have to know the particular host on which a service is implemented. The user should also not have to know that a service is implemented on several machines for reliability or load sharing.

Second, the generation and manipulation of names by users should be easy. A naming system at least as powerful as the Multics directory hierarchy is needed to allow users to select names without fear of conflict.

Third, the naming system should be resilient to single hardware failures. Any individual system or communications link going down should not bring down the naming system. This precludes the use of a single "naming computer."

Fourth, the naming system should continue to function even after major portions of the network are down. If a service is reachable through the network, then it should be nameable, even if all the rest of the network (outside of the path to the service) is down.

8.1.2 Basic Implementation

Reed's naming scheme consists of a directed graph whose vertices correspond to either directories or generic services. Directories have zero or more child edges that point to other vertices. Services do not have edges emanating from them. Each edge emanating from a directory has a character string name. There is a directory called the root from which all other vertices may be reached. A sequence of names from the root to a particular vertex is called the treename of the vertex. Note that multiple edges may point to the same vertex, and therefore, loops may exist in the naming hierarchy. (1) Each directory will be implemented as a list of names of edges and associated port identifiers (2) that implement the target vertices. The target vertex of an edge need not reside on the same host as its parent.

---

(1) The edges in Reed's naming scheme constitute so-called "hard links" to objects, because each edge points to an actual vertex. A naming scheme can also include "soft links" that merely consist of character string treenames. In interpreting a name, if a "soft link" is reached, the character string name contained in the "soft link" replaces that portion of the original name from the root to the "soft link" that has already been interpreted. Interpretation is now restarted at the root with the new name. Thus, "soft links" are merely indirect pointers to new names and do not have security relevance. "Soft links" correspond to links in the Multics directory hierarchy.

(2) Ports were defined in chapter 7.

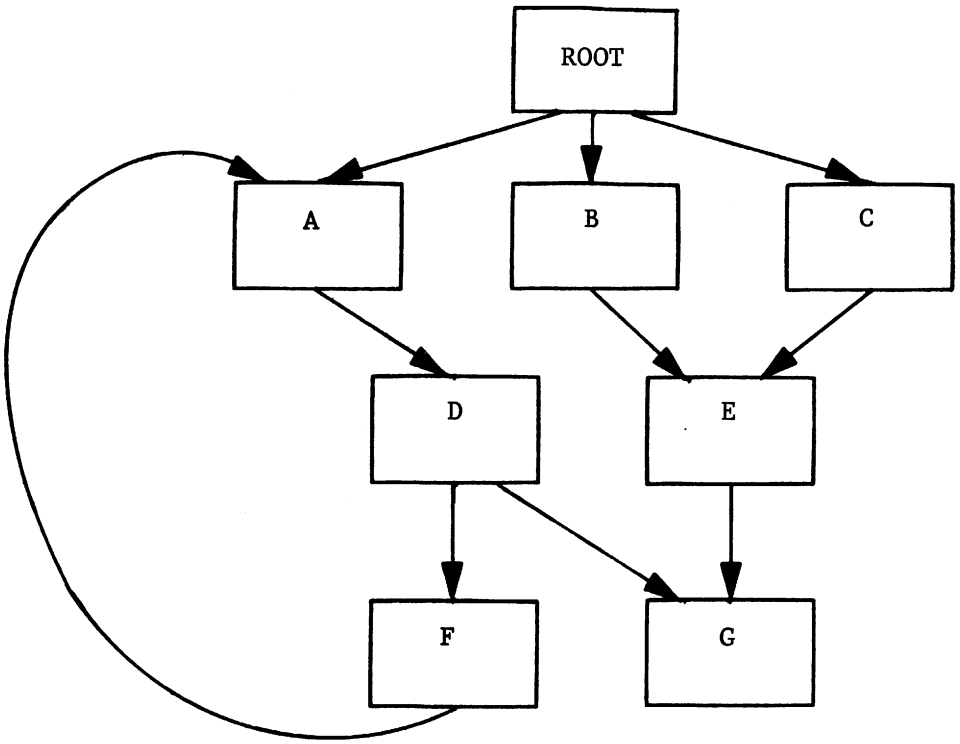


Figure 8.1 Typical Naming Network

Figure 8.1 shows an example of a naming network. Service G can be reached by a variety of treenames. Treename interpretation occurs as follows:

1. The user sends the treename ROOT.A.D.G in a message to the root.

2. The root strips off its own name and looks up the port number for A. The root then sends A a message consisting of the treename

A.D.G, the treename that was used to reach A, namely ROOT, and the port number of the user process.

3. A performs the same operation and sends D a message consisting of D.G, ROOT.A, and the user's port number.

4. Finally D sends a message to G consisting of G, ROOT.A.D, and the user's port number.

5. G now responds to the user's port to initiate a network connection to provide the required services.

As can be seen from the example, each directory strips off its own name, looks up the next edge, and sends a message to the child vertex. The treename evaluated so far is passed along for optimization purposes. If a user frequently requests the same or very similar treenames, it would be inefficient to re-evaluate the treenames from the root every time. Therefore, each directory can (optionally) send its own port number and the treename evaluated so far back to the user. Reed demonstrates that the user can maintain an associative memory of partially evaluated treenames that recognizes changes in the directory structure that invalidate earlier entries in the associative memory. The scheme is not reproduced here, because it is not impacted by security requirements.

8.1.3 Garbage Collection

Reed proposes a scheme for garbage collection when the ports associated with vertices are destroyed. Reed defines a timeout for each edge in the naming network. Unless the timeout is extended, the edge is deleted when the timeout expires. In the simplest case, a vertex can extend its own timeout as long as it wishes to remain in existence.

8.1.4 Other Topics

Reed discusses a number of other topics related to his naming scheme including approaches for robustness under loss of major portions of the naming network, additional optimization strategies, and details of the implementation on the proposed M.I.T. Laboratory for Computer Science Local Network. These issues do not have a major security relevance and are not discussed in this thesis.

8.2 Incorporating the Lattice Model in the Naming Scheme

Reed's naming scheme explicitly did not address protection issues. In this section, we shall examine the issues of incorporating the lattice model into the naming scheme. We shall see that the lattice model fits into the scheme quite smoothly, but we shall also see that Reed's garbage collection strategy cannot be implemented under the lattice model. We shall also see that the underlying implementation of the network naming scheme in terms of a host's local naming scheme can have adverse affects on the overall scheme.

The basic approach to incorporating the lattice model will be similar to the approach taken in the Case model <Walter74>. Each directory and service vertex will be assigned an access class. Operations on vertices will then be defined to preserve the simple security and confinement properties. Two major new issues must be considered that were not present in the Case model, which was based on the Multics directory hierarchy exclusively. First, Reed's naming scheme allows a vertex to have multiple parents. Since previous naming systems that have incorporated the lattice model have had only single parents, multiple parents could be expected to have major security implications. However, we shall see that multiple parents can be accommodated in the lattice model without major difficulties. Second, the allowable list of operations could differ depending on whether a vertex is implemented on a dedicated host or a multilevel host. In this case, we shall see that, indeed, the list does differ.

8.2.1 Notation

In the remainder of this section, the following notation will be used. An untrusted process P is making requests on the naming network. The access class of P is denoted by AC(P). Since P is untrusted, it may contain a "Trojan Horse". Multilevel hosts may also have a trusted process TP. Vertices V1, V2, and V3 have access classes AC(V1), AC(V2), and AC(V3). The three vertices are implemented on hosts H1, H2, and H3. The process P is implemented on host HP. The hosts may be either multilevel or dedicated as indicated in the text. The maximum access class of a host (or the dedicated access class) is denoted by AC(H1), or AC(H2), etc. Access classes form a partial ordering in which two access classes may be <, =, >, or disjoint.

8.2.2 Creating Edges and Vertices

For an untrusted process P (which presumably could contain a "Trojan Horse") to create an edge from vertex V1 to V2, where V2 is a new vertex, P must write in directory V1. Therefore, $AC(P) \leq AC(V1)$. P must also read V1 to detect name duplication errors. Therefore, $AC(V1) \leq AC(P)$. These two constraints together force $AC(V1) = AC(P)$. Since V2 is being created for the first time, P is effectively writing at the access class of V2. Therefore, $AC(P) \leq AC(V2)$, and $AC(V1) \leq AC(V2)$.

The constraint AC(V1) $\leq$ AC(V2) is called the compatibility property by Bell in <Bell74>. In the Multics directory hierarchy, an object has one and only one parent. Therefore, compatibility must be enforced throughout the hierarchy, and the access class of a parent directory must always be less than or equal to any descendent. However, Reed's naming scheme allows multiple parents and naming loops. Assume we have three vertices V1, V2, and V3, where AC(V1) $\leq$ AC(V2) $\leq$ AC(V3). Assume V1 is a parent of V2, and V2 is a parent of V3. If compatibility must hold, then V3 cannot be a parent of V1 unless AC(V1) = AC(V2) = AC(V3). Since this is an unreasonable constraint on the choice of access classes, this seems to preclude one of the generalities Reed desired in his naming scheme, the ability to have loops.

Fortunately, compatibility is required only when creating vertices. If one wishes to define an edge from V1 to V2, where V1 and V2 already exist and not(AC(V1) $\leq$ AC(V2)), the only requirement is that AC(P) = AC(V1) as before, and AC(V2) $\leq$ AC(P) so that P may know of the existence of V2. Thus, at least for defining edges between existing vertices, the compatibility constraint need not be enforced.

In simpler terms, when adding a vertex to the naming network, the first edge to be added must comply with the compatibility property, i.e., the access class of the first parent must be $\leq$ the access class of the new vertex. However, once the first parent has been created, additional parent edges may be added without complying with the compatibility property.

8.2.3 Using Directories and Services

In the previous section, we showed how P could use (and modify) directory V1, if AC(P) = AC(V1). Now assume we have directory V1 with an edge pointing to service V2. Assume AC(V1) < AC(V2) and AC(P) = AC(V2). Ostensibly, P should be able to use V2, because the access classes are equal. However, for P to reach V2 through the naming system, P must send a message to V1. (1) However, since AC(V1) < AC(P), the confinement property forbids P sending a message to V1. If V1 is implemented on a dedicated host H1, then AC(H1) = AC(V1) < AC(P), and the confinement property restriction must hold. In that case, if V1 is the only parent of V2, then neither P nor any other untrusted process at any access class can ever get to V2. However, if H1 is a multilevel host and AC(P) $\leq$ AC(H1), then P can use V1 as follows: When P's message arrives at H1, it must be fielded by a trusted process TP. TP is guaranteed not to contain a "Trojan Horse". TP notes the access class of P's message and creates an untrusted process Q such that AC(Q) = AC(P). Q can now read the contents of V1, determine the port number of V2, and complete the directory search operation. Since Q is an untrusted process, it is forbidden by the confinement property to write in V1. (2)

---

(1) Section 7.2.2 discussed the origin of this problem.

(2) One additional point must be noted. Directories are normally shared by many processes, and therefore synchronization is needed to assure consistency. Section 8.3 discusses a scheme for synchronization without writing.

It should be noted that this scheme for searching a directory at a lower access class is also useful for using services at a lower access class than the requesting process. For example, a multilevel host H1 may offer a PL/I compiler service at the lowest access class (unclassified). If AC(PL/I) < AC(P), then process P cannot send a message to the process implementing the PL/I service. However, if AC(P) $\leq$ AC(H1), then P's request can be fielded by a trusted process TP that creates an untrusted process Q, such that AC(Q) = AC(P). Q can now compile P's PL/I program, since Q can have read-execute-only access to the code that makes up the PL/I compiler. By creating Q, we have changed the read-write operations of sending messages back into the read-only operation that running the compiler would have been, had the system not been decentralized.

There is no fundamental reason why there must be multiple trusted processes TP and untrusted processes Q for each service instance. For efficiency reasons, a host could choose to multiplex a single trusted process TP and a set of untrusted processes Q, with one Q per access class currently in use. Q's need not exist for every access class that might ever be used. Rather, TP can create Q's at the desired access classes as needed.

8.2.4 Explicit Deletion of Edges and Vertices

Deleting edges and vertices from the naming network again leads us to questions concerning the compatibility property. These questions arise, because we wish deletion to occur cleanly, without leaving

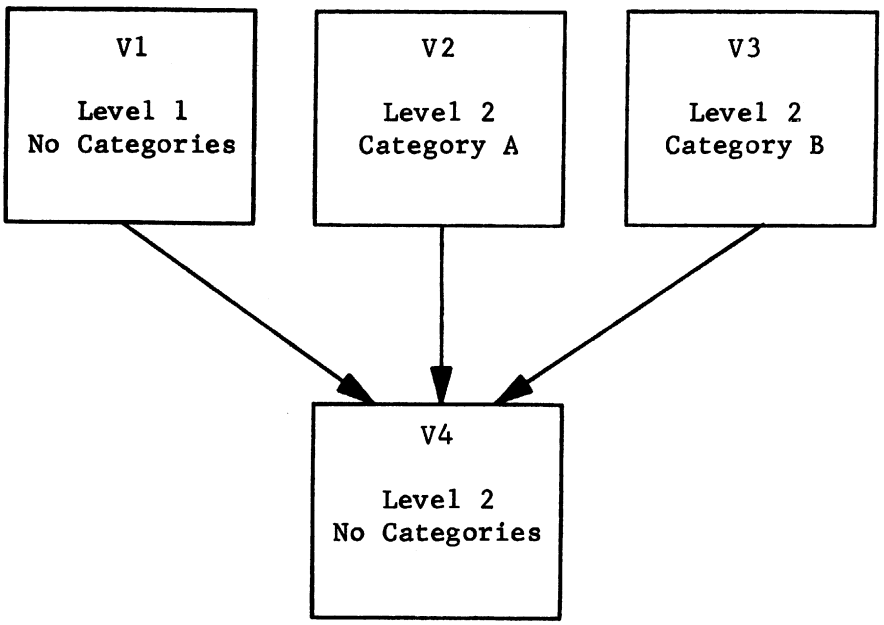


Figure 8.2 Deletion Example

so-called "lost objects" in the naming network. Figure 8.2 shows a portion of a possible naming hierarchy that we shall consider in this section. V1, V2, and V3 are all parents of V4. AC(V1) < AC(V4), which

maintains compatibility, but AC(V4) < AC(V2) and AC(V4) < AC(V3), which do not maintain compatibility. Note that AC(V2) and AC(V3) are disjoint, although both are > AC(V1). We assume that the desired behavior of the naming system is that when all parent edges of V4 are deleted, then V4 is also deleted.

For process P to delete the edge V1.V4, P must write into V1. Therefore, AC(P) $\leq$ AC(V1). For P to know of the existence of V1.V4, P must read V1. Therefore, AC(P) = AC(V1). P cannot delete V4 based solely on the deletion of V1.V4. P does not have the right to know whether V2.V4 or V3.V4 exist, but P must assume that they might exist. P must somehow tell V4 that one of its parents has been deleted. If V4 maintained a count of parents, then V4 could delete itself when the parent count went to zero. (1) Presumably, P sends a message to V4 when V1.V4 is deleted. However, if another process Q wishes to delete V2.V4, a problem arises. We must require that AC(Q) = AC(V2), for the same reasons that we required AC(P) = AC(V1). However, this implies AC(Q) > AC(V4), and the confinement property rules would forbid Q sending a message to V4. If H4, the host on which V4 is implemented is a dedicated host, then the confinement property must be enforced, and V4 cannot be deleted. But, if H4 is a dedicated host, the edges V2.V4 and V3.V4 are useless for any type of access to V4. Therefore, the only interesting case is if H4 is a multilevel host.

---

(1) This parent count mechanism is adapted from a similar mechanism in the Cambridge University CHAOS system <Slinn76>.

If H4 is a multilevel host, and if AC(V2) $\leq$ AC(H4), and if AC(V3) $\leq$ AC(H4), then Q can send the message to the trusted process TP that implements V4 on H4. TP can maintain the parent count and delete V4 when the count goes to zero.

Unfortunately, we have overlooked a basic difficulty. If V3.V4 is the last parent edge of V4, the deletion of V4 is itself an operation that contains information at AC(V3) which may be visible at AC(V4). Even though the trusted process TP performs the deletion, TP is doing so, only because process Q at access class AC(V3) requested it. Since AC(V4) < AC(V3), this could constitute a confinement property violation if the deletion were visible at the access class of V4.

Whether the deletion of V4 is visible at the access class of V4 is determined by the implementation of the network naming scheme in terms of the local naming scheme of H4. We will examine several possible implementations of local naming and evaluate their impact on this problem.

8.2.4.1 Multics Style Naming

Multics very strictly enforces the compatibility property, because each object has only one parent. Therefore, the deletion of V4 must be visible at the access class of V4. The trusted process TP could attempt to copy V4 into a new segment at a higher access class when V1.V4 is deleted. In this way V4 could seem to be deleted as soon as V1.V4 was

deleted. However, since AC(V2) and AC(V3) are disjoint, there does not exist a common access class to which V4 could be upgraded that would not receive information in violation of the confinement property if either V3.V4 or V2.V4 were deleted. Therefore, V4 cannot be deleted when the last parent edge is deleted. Note that V4 is not a "lost object" if all its parent edges are deleted. It can still be accessed through the local Multics hierarchy, and could be deleted by a human being making an explicit decision to downgrade the information that no parent edges existed any longer.

8.2.4.2 CAL Style Naming

The CAL system <Lampson76> allowed objects in the hierarchy to have multiple parents. However, one of the parents was always distinguished as the "owner" of the object, and deletion of the object was determined by the "owner" parent. Therefore, although CAL allowed multiple parents, the "owner" characteristics would force CAL to have the same deletion difficulties as Multics.

8.2.4.3 Naming With Revocable Capabilities

If directories are implemented on a capability - based system as lists of revocable capabilities, (1) then the problem of deletion is

---

(1) Redell <Redell74> describes the implementation of revocable capabilities.

solved. Each time TP receives a request for access to V4, it grants the access as a revocable capability. A separate capability is created for each access class that is requested, so that TP can selectively revoke by access class. Now when P requests that V1.V4 be deleted, TP revokes all capabilities for V4 except those that were derived from V2 or V3. Thus, all capabilities except those for which AC(V2) $\leq$ AC(capability) or AC(V3) $\leq$ AC(capability) are revoked. Once their capabilities have been revoked, any process at a lower access class can no longer determine the existence or non-existence of V4. Such a revocable capability naming scheme could be implemented on the HYDRA kernel <Cohen75>.

8.2.4.4 UNIX Style Naming

In the UNIX operating system <Ritchie74>, each object is allowed to have multiple parents, and there is no distinguished "owner" parent. If a security kernel were implemented for UNIX (1) in which UNIX files were mapped as segments in the PDP-11/45 name space, then an equivalent function to the capability revocation could be performed by TP when a parent edge is deleted. TP could request the kernel to do a "setfaults" operation (2) on V4 after V1.V4 was deleted. As a result of the

---

(1) Security kernels for UNIX are presently under development at the MITRE Corp. <Biba77> and at U.C.L.A. <Kampe77>.

(2) "Setfaults" is a term taken from the Multics operating system. "Setfaults" sets fault bits in the segment descriptor words for a particular segment in the descriptor segments of all processes that currently have the segment mapped. "Setfaults" is used to force all processes to re-establish their access rights to a segment, for example, after an access control list has been modified.

"setfaults", all processes would be forced to re-establish their access to V4, and only those with access to V2 or V3 would succeed. Thus, a security kernel for UNIX could also allow V4 to be deleted automatically.

8.2.5 Garbage Collection

Reed's strategy for garbage collection based on timeouts can be implemented only on multilevel hosts. If we have two vertices V1 and V2, and V1 is the parent of V2, and AC(V1) < AC(V2), then V2 cannot send messages to V1 extending its own timeout. If H2 is a multilevel host, then a trusted process TP could extend the timeouts, but only if AC(H2) $\leq$ AC(H1).

At present, there is insufficient experience with decentralized computing systems to determine if the lack of a garbage collector is a serious restriction. Experiments with and without garbage collectors will be required to determine actual system requirements in this area.

8.3 Synchronization Without Writing

In section 8.2.3 we identified the need for synchronization between untrusted processes at different access classes. Assume a shared data base exists and is modified by processes at a low access class, and there exist reader processes at a high access class who wish to read the data base, assuring consistency without sending information to the low access class processes. Dijkstra semaphores <Dijkstra68> are inadequate for this task, because two-way communication can be implemented with P and V operations. Semaphores are inherently read-write objects to all users.

Reed and Kanodia <Reed77> propose a scheme for process synchronization using eventcounts that allows synchronization between processes at different access classes without violation of the confinement property. An eventcount is a non-decreasing integer variable on which two operations are allowed - read and advance. The read operation returns the value of the eventcount. It does not modify the eventcount in any way, and thus cannot be used to transmit information. The advance operation adds one to the value of the eventcount. Both read and advance must occur indivisibly.

Assume we have a data base shared between two processes. One process is the writer process and operates at the access class of the data base. The other process is the reader process, and it operates at a higher access class. Two eventcounts called in and out are defined. The writer process advances in, updates the data base, and then advances

out. The reader process reads in, waits for the value of out (determined by reading out) to equal the value read from in, and then reads the data base. The reader now reads in again. If the value of in has changed, then the data base may have been written while the reader was reading, and the data extracted may be inconsistent. In this case, the reader process must go back and retry the entire operation.

While this solution potentially requires the reader process to repeat some work, it allows synchronization without violating the confinement property. The solution described here is a simplified version of Reed and Kanodia's, and it works for any number of simultaneous reader processes and exactly one writer process. A solution for multiple writers is shown in <Reed77>, but was omitted here, due to its additional complexity. However, the multiple writers solution also preserves the confinement property. An equivalent solution was developed independently by White <White75>.

Chapter Nine

Downgrading Information

Downgrading of information is the process by which information classified at a particular access class is reclassified at a new lower access class. In the "paper world," downgrading of information occurs for two major reasons. First, a human being may read a classified document, extract an unclassified paragraph, and reprint it elsewhere. Second, information classified for national defense reasons has mandatory downgrading times specified by Presidential Executive Order 11652 <Nixon72>. After a specified period of time, any classified document (with a few exceptions) must be downgraded to the next lower access class, and eventually, must be declassified completely.

Because of the threat of "Trojan Horses," we cannot allow uncertified computer programs to perform either type of downgrading. This no-downgrading restriction is formalized as the confinement property. In this chapter, we shall explore the needs for computerized downgrading of information, and we shall see how distributing the downgrading functions among different processors can make the function easier to certify secure.

9.1 Why Downgrade Information?

Downgrading information in a computer system may be desirable for two major reasons. First, files may be downgraded due to statutory or other requirements. This type of downgrading does not have a time criticality associated with it, and therefore can be accomplished by a trusted user such as a system security officer (SSO). This type of downgrading has been examined in depth in <Whitmore73> and <Schiller76> and will not be discussed here. The other major reason for downgrading is to provide "sanitized" versions of information. Sanitization generally involves extracting selected portions of the classified information that can be released at lower access classes.

Sanitization often has a high degree of time criticality associated with it. If a user has constructed a file that he or she knows is unclassified, then the system should mark it as such. The user does not want to talk to the SSO, every time he or she wants to send mail at a lower access class. An example of a more serious timing constraint on sanitization is found in military intelligence centers. An intelligence analyst may be processing some highly classified reports. Based on his data correlations, the analyst determines that an enemy is about to attack. That information must be quickly sanitized and released to an operational commander, if effective countermeasures are to be taken. The commander sometimes cannot be given direct access to the intelligence reports, because that could compromise the intelligence

sources. Therefore, rapid sanitization of the data is essential. (1)

Sanitization is also important in civilian applications. For example, a corporate executive may wish to generate a sanitized business projection from a highly sensitive long range projection data base. Such a sanitized projection might be required to respond to government queries, or, on a more routine basis, to generate next month's production schedule.

Sanitized information may pass between processes on multilevel systems, but the more important application is for sanitizing information to be passed between dedicated host systems. If we can effectively sanitize messages, we have eliminated many of the drawbacks of one-way communication that were described in chapter 7.

---

(1) Examples of the need for this type of sanitization to protect intelligence sources can be found in <Cave Brown75>.

9.2 Formularies

Stork <Stork75> describes a technique for downgrading information based on preprogrammed sanitization criteria. If one could write a program that could examine messages to determine if they were properly classified, that program (called a formulary by Hoffman <Hoffman70>) could mediate the downgrading of information and allow a limited form of two-way communication between hosts operating at different dedicated access classes. The formulary could run in a trusted process on a security kernel based processor interposed in the communications path, or the formulary could run on a dedicated certified minicomputer.

Formularies are very difficult to implement in general. They must make a classification decision based on the data present in a message, yet the message may have an arbitrarily complex interpretation. For example, if the message to be sanitized is the output of an electronic mailing system, then the formulary presumably would require a natural language understanding capability. Since general natural language understanding is beyond the current state of the art, at least some set of formularies are presently infeasible. Worse still, the formulary must be resistant to attack by "Trojan Horses." The formulary must be able to distinguish any classified information anywhere in the message. Since the "Trojan Horse" could encode information using an arbitrarily complex scheme, the feasibility of a formulary discerning the attack is questionable at best. For example, if the data to be sanitized is a table of 10-digit numbers, the "Trojan Horse" could encode information

by modifying the low order digits. As long as the formulary did not check for a high degree of accuracy (which could be impossible if the purpose of the untrusted "Trojan Horse" program was to compute the 10-digit numbers for the first time), such communication could go completely undetected. Nibaldi describes this type of attack on formularies by "Trojan Horses" <Nibaldi76>.

One type of formulary that can be implemented without a "Trojan Horse" threat is one that automates the statutory downgrading procedure. If every file is marked by the security kernel with its creation date and downgrading schedule, a fully automated formulary could perform the scheduled downgradings without vulnerability to "Trojan Horse" attacks.

One very simple to implement formulary is one that refers its sanitization decision to a human being. All the formulary must do is present the data to the human being and await the human's response. Since human beings have a relatively sophisticated natural language capability, they can usually do much better than the formulary programs at understanding the content of a message. Human sanitization, of course, is what is used in the "paper world" for all downgrading decisions. A human review formulary must still be supported by a trusted process. Otherwise, the system could not assure that the human's decision would be implemented. One such human review formulary has been implemented by the MITRE Corp. in a security kernel demonstration system <Mack76>. In the demonstration scenario, an intelligence officer in an air defense center selectively downgrades classified air tracks, so that an uncleared operations officer may see upcoming threats and direct defensive forces to respond.

Human review introduces obvious limitations in the bandwidth of communications, but may be acceptable for many applications. Human beings are also subject to "Trojan Horse" attacks. Humans are also vulnerable to obscure encodings of information in the messages. For example, a "Trojan Horse" could encode information in non-printing characters between legitimate words of a message. Since the non-printing characters are not displayed, the human reviewer would be unaware of their presence. The next section describes some approaches to aid the human reviewer, primarily through the use of secure intelligent terminals.

It must be emphasized that all one can accomplish with human review is <u>better</u> security, never <u>perfect</u> security. The human will always make mistakes, and the "Trojan Horse" will always be able to sneak some things by.

9.3 Secure Intelligent Terminals

In this section, we shall examine the potential of using multilevel secure terminals to alleviate some the difficulties associated with human sanitization of information. We shall assert that by moving some of the downgrading functions into an intelligent terminal, (1) we can improve the human engineering of the sanitizing operations, and can more easily verify the correct implementation of the security related functions.

9.3.1 User Requirements

From the user's point of view, a secure downgrading terminal should offer a number of features to assure both convenience of use and effective security. First, the user should be able to establish several simultaneous network connections at different access classes. These connections should be possible to both dedicated hosts and to untrusted processes on multilevel hosts. The terminal must assure that data from different connections is not mixed, since any of the processes to which the terminal is connected could have a "Trojan Horse."

---

(1) By an intelligent terminal, we mean something far more capable that the current commercial "intelligent" terminals that can do no more than simple character insertion and deletion in a small buffer memory. An intelligent terminal, as used here, would have a significant processing capability and a reasonably large local memory. Such an intelligent terminal would be able to store complex data structures, display them for the user, and interact with the user in a sophisticated and dynamic way to update and edit those structures, with little if any interaction with the host computer. Examples of such intelligent terminals include the IMLAC PDS-4 and DEC GT-40 graphics display terminals and the new Xerox Soft Display Word Processor (SDWP) <Shemer77>.

The terminal should be capable of subdividing the display screen into small windows that can be separately controlled by each network connection. Windows should be of variable size, and the terminal should assure that a network connection is confined within its particular window. Ames discusses the needs for multiple windows in <Ames76>.

Security labelling should be clear and reliable. The user should be able to determine at a glance the access class of each window. (1) The sophisticated user will want to write programs to run in the terminal to interact with software on the central host. Therefore, untrusted code supplied by a "Trojan Horse" should not be able to overwrite security labels displayed in "protected fields," nor should it be able to create false security labels. The user should be able to tell at a glance to which network connection the keyboard is currently attached. The user should be able to look at an access class display that is integral to the keyboard to determine the current access class.

The multilevel terminal should have a controller mode in which the user can open and close network connections, allocate display windows, and determine status information about the network connections, etc. Since the controller functions must cross access class boundaries, they must be implemented in the terminal by a trusted process.

---

(1) Note that the name of an access class may include both a level name and a set of several category names. The terminal must allow sufficient space for the complete access class name, although some types of abbreviation may be used.

Downgrading of information in a multilevel terminal would consist of identifying a candidate piece of text in a window at one access class, passing the text to the multilevel controller, and verifying that the text has been properly sanitized. The user should be provided a pointing device such as a lightpen or a mouse <Engelbart68> to identify the text to be downgraded. The candidate text should be highlighted in its window, and then redisplayed by the multilevel controller in another window. By using the redisplay technique, the user can verify that the text passed to the multilevel controller by uncertified software is the same text to which the user pointed with the mouse. The user should be able to approve the downgrade by pushing a single downgrade button. The terminal should keep a log of all downgrade requests to assure human accountability. The log could be kept on another host on the network, stored at the highest access class that is processed by the terminal. The log does not provide a direct security control, but does provide a mechanism for checking on the user who may make unwise or unauthorized downgrade requests.

9.3.2 Implementation

Implementation of the type of multilevel terminal described here is envisioned on a machine similar to the Xerox SDWP <Shemer77>. There are three major aspects of the implementation that have security implications - the processor and memory configuration, display windowing, and physical protection.

9.3.2.1 Processor and Memory Configuration

Since the software for a terminal like the SDWP is going to be large and complex, it will be impossible to verify its complete correctness. Therefore, the multilevel terminal will require a security kernel.

In a few years as LSI technology progresses, a machine like the PDP-11/45 or the Honeywell Secure Communications Processor (SCOMP) <Broadbridge76> will be available on a single board. These machines have the descriptor based addressing and multiple protection states needed to run security kernels. Using such a very inexpensive machine would allow each terminal to run a security kernel supporting a moderate number of untrusted processes, one per network connection. The security kernel (or a trusted process) would handle the network interface, routing messages to each of the untrusted processes. The security kernel process would also allocate the keyboard and pointing device from process to process as requested by the user. However, there would always be a button on the keyboard to allow the user to communicate directly with the kernel. The button would be read only by the kernel, so that other processes that might be running "Trojan Horses" could not masquerade as the kernel.

9.3.2.2 Display Windowing

Breaking up the display into windows of dynamically varying size is a difficult task on a convention display. If the display were assigned directly to an untrusted process, then a "Trojan Horse" could overwrite other windows and the security labels. If the display were assigned to the kernel, then the kernel would have to interpret all display commands to ensure that the window boundaries were observed. However, such software interpretation of display commands can be very slow and extremely difficult to verify correct.

To overcome these difficulties, a concept is borrowed from the descriptor based processes to create what we shall call "descriptor based displays." The Xerox SDWP uses a bit map raster scan display driven by a display processor <Hartke77>, independent of the main processor in the terminal. If the display processor implemented two descriptors for accessing the bit map - an x-descriptor and a y-descriptor, (1) then the security kernel could load the descriptors with the x and y dimensions of the desired window and assign the display directly to the untrusted process. The display processor would assure that the bit map was addressed only within the limits of the x and y-descriptors. This technique is analogous to the use of descriptor based I/O in the SCOMP. Descriptor based I/O allows the user to directly control I/O channels, because the channels accept only virtual

(1) Each descriptor would contain an upper and a lower bound for that particular dimension of the window.

addresses from the user. Analogously, a descriptor based bit map display allows the user to specify only virtual addresses within the bit map.

9.3.2.3 Physical Protection

Multilevel secure terminals have particular physical protection requirements. First, the terminal must be protected against physical tampering. Otherwise, the security kernel could be bypassed by surreptitiously modified circuitry. Second, storage media within the terminal must be erased after use. Such erase requirements may be very time consuming if the terminal includes a large, but slow disk. (1) Third, the terminal must store and protect cryptographic keys for communication on the network. If end-to-end encryption is used, multiple keys must be stored. A more detailed consideration of the physical security requirements for multilevel terminals can be found in <Gardella76>.

(1) Alternatively, the disk (and other storage media) could be removed from the terminal and stored in a vault.

Chapter Ten

Administration of the Lattice Model

Use of the lattice security model in large decentralized computing systems introduces two administrative issues not present in centralized systems. First, although individual hosts will generally use only a small number of access classes, the network as a whole will use a large number of usually disjoint access classes. Second, as host systems join the network and new applications are created, new access classes must be defined. The network must provide a mechanism for conveniently defining new categories and assigning clearances for them without having to use a central Network Security Officer for every operation.

10.1 Proliferation of Access Classes

Individual hosts on a network in general use only a small number of security categories to separate information. For example, the Honeywell Multics system supports eight sensitivity levels and up to eighteen categories <MPM75>. The eighteen categories are represented as an eighteen-bit array, so that any combination of the eighteen categories may be represented. Although this means that Multics potentially supports 8 sensitivity levels times 262,144 category combinations for a total of 2,097,152 access classes, in fact only a small portion of those combinations are ever used, because most files can be assigned to one or

at most a combination of two categories. Very few files contain information from several categories.

As a network grows, however, the number of categories required will also tend to grow. Each host is likely to have some particular collection of sensitive data to be compartmentalized that does not match any already defined categories. If each host contributes only five categories, a network of only twenty hosts must support 100 categories for a total of 10,141,204,801,825,835,211,973,625,643,008 (which is 2**103) access classes. Clearly this type of exponential proliferation must be limited.

Since only a small number of categories are in use at any one time, the network can create and destroy processes dynamically in response to different access class demands. Therefore, although the network could support many millions of access classes, it would have to support at most one access class per packet transmitted, and therefore at most one untrusted process per packet. This is of course a worst case, since a network connection at a given access class will typically exchange many thousands of packets at the same access class.

However, we still have a problem representing the security labels in a network that supports a large number of categories. If the network supports 1000 categories (a not unreasonable number for a nation-wide commercial network that protects personal information with categories), then if the Multics technique of assigning one bit per category is used to allow representation of all possible combinations, then the header of

a packet must store 1000 bits just for the security label. The packet size for some networks has been suggested to be as small as 128 bits <Danet76>. In that particular case, security labels would require an 800% overhead. (1) Even if the 1000 bits could be tolerated in communications, the 1000 bits must be stored with every object on every host in the network.

To reduce these types of storage consumption, we propose to take advantage of the fact that very few categories are used in combination at any one time. If each category is assigned a unique 32-bit number, (2) then an access class could be represented as a sensitivity level, an integer indicating the number of categories to which this object belongs, and a list of the category numbers. If we support eight sensitivity levels (3 bits) and use 29 bits for the integer number of categories, then an object residing in only one category requires only 64 bits for its security label.

An even more compact representation is possible if unique numbers are assigned to frequently used category sets. Assuming a 3-bit level number and a 29-bit category set number, an access class could be stored in only 32 bits. If category sets exist for which category set numbers have not been assigned, but for which category numbers have been

---

(1) This is assuming that every packet requires a security label. In the case of link encryption, this is true. For end-to-end encryption, the label could be implicit in the cryptographic key used to encipher the data.

(2) A 32-bit category number is sufficient for approximately twenty categories each for every man, woman, and child in the United States of America.

assigned, then a special category set number could be reserved to mean that the list of categories notation described above is being used.

For efficiency reasons, a host may wish to translate a small number of frequently used category numbers into the bit array representation described above. This translation could be performed independently at each host, as long as all network traffic used the standard labelling.

10.2 Assignment of Categories and Clearances

The lattice security model, by its very nature, suggests a centralized authority for handling category definitions and for assigning clearances. The non-discretionary nature of the lattice model says that the individual user may not make these decisions. In a single host, a system security officer (SSO) is responsible for assigning access classes and clearances. (1) However, in a network with decentralized authority, each host will have an SSO responsible for that particular host. The SSO's will want to periodically define new categories, assign clearances to users and to other hosts, etc. It is not acceptable that there be a network SSO to perform all of these tasks, because far too much time loss would be introduced into the system. Therefore, we make use of Branstad's concept of a Network Security Center <Branstad73> to provide an automated way to assign new categories and clearances.

---

(1) <MAM76> describes the SSO functions for Multics.

When an SSO wishes to define a new category, he or she sends a message to the NSC requesting the category. Because the SSO functions run in a trusted process, the NSC can be assured that no "Trojan Horses" will interfere. The NSC assigns a 32-bit category number and returns it to the SSO. At the same time, the NSC marks the host on which the SSO is running as cleared to receive that category. Marking a host cleared presumably involves broadcasting that information to relevant communications processors in the network. Now the SSO can assign clearances within his or her own host.

An SSO may also wish to grant clearances to other hosts on the network for categories to which he or she has access. This can also be accomplished by sending a message to the NSC, which then authorizes the appropriate hosts. The NSC should only allow an SSO to grant access to categories to which he or she is already authorized. The SSO can be trusted because:

a. The SSO is already authorized the category, and

b. The SSO functions run in a "Trojan Horse" - free trusted process.

Since the SSO is limited to granting clearances for access classes to which he or she already has access, the SSO is limited from doing any serious damage beyond that already possible.

[This page intentionally left blank.]

Chapter Eleven

Limitations of End-to-End Encryption

Throughout the rest of this thesis, it has been assumed that all communications have been encrypted to counter the threat of electronic eavesdropping through either wiretapping or listening to radio broadcasts. Two types of encryption were discussed in section 2.2.2, link encryption and end-to-end encryption. Link encryption has often been criticized, because all packets must pass through the network communications processors in plaintext. Therefore, it has often been claimed that end-to-end encryption was the preferable encryption technique for decentralized computing systems. Not only does end-to-end encryption ensure that communications processors see only ciphertext, but there is a general category of communications networks, broadcast networks, for which link encryption is inappropriate. Thus, one could easily conclude that end-to-end encryption is an ideal solution.

Unfortunately, we shall show in this chapter that end-to-end encryption, far from being a panacea, can provide a "Trojan Horse" with an effective means of surreptitious communication. Thus, we shall show that link encryption is still necessary for many applications. For broadcast networks, where link encryption is inappropriate, the technique of dynamic key renaming is proposed to block the "Trojan Horse" threat for this particular class of end-to-end encryption.

11.1 The Problem

To understand the basic problem with end-to-end encryption, it is necessary to establish a scenario under which attack can occur. Two host computers exist between which two processes are communicating. Both hosts have security kernels, and therefore can be assumed to correctly label each outgoing packet with its proper security label. The two processes are untrusted, and therefore are operating at the same access class. For convenience, one will be designated the sender process and the other, the receiver. Since the processes are untrusted, both may contain "Trojan Horses."

Between the two hosts is a network of communications processors. A large number of other hosts are also connected to the network, and multiple paths exist between any pair of hosts. The communications processors dynamically route packets through the network, selecting optimal paths based on changing traffic levels. The communications processors do not have security kernels and may contain "Trojan Horses." Each host is equipped with end-to-end encryption hardware located between the host and the local communications processor. Because the communications processors see only ciphertext, link encryption is not used.

The scenario just described would seem to be secure, because all data is encrypted before being passed to the communications processors. However, certain control information must be passed in cleartext from the host to the communications processor to allow the network to

function. This control information consists of the destination address for the packet, the length of the packet, and the time between successive packet transmissions. (1) All three of these control signals can be controlled by the untrusted user process. Therefore, a "Trojan Horse" in the sending process could encode information in these control signals that would be readable both by the untrusted software in the communications processors and by anyone tapping the communication links. The external wiretapper can be excluded by adding link encryption, but the "Trojan Horse" in the communications processor can still receive the information, fabricate a new packet, and route it to some other host, not authorized the access class of the sender and receiver processes.

One could assert that the bandwidth available in this way is insufficient for the "Trojan Horse" to effectively communicate. However, one estimate in <Padlipsky77> suggests that over 100 bits per second could be passed with the destination address alone. Most teletype communication in the world occurs at 75 bits per second. Clearly, the "Trojan Horse" communicating in control signals is a significant threat.

---

(1) The time between successive packets is passed implicitly.

11.2 Countermeasures

The most obvious countermeasure to a "Trojan Horse" signalling via control information is to place a security kernel in every communications processor and use link encryption. Now any "Trojan Horses" remaining in the uncertified code of the communications processors are effectively confined by the security kernels and cannot leak any information, and the link encryption stops the wiretappers. However, now one can question why use end-to-end encryption at all. Since it provides no additional security, one could eliminate the end-to-end encryption and reduce the expense and complexity of the system.

However, end-to-end encryption must still be considered for two reasons. First, two users of a shared communications processor may not be willing to trust either the certification of the security kernels or the persons with physical control of the processor. Ford and Chrysler would not trust a shared communications processor. Second, we have not considered broadcast networks in which link encryption is inappropriate. Therefore, the next three subsections deal with closing each of the control signal paths - packet length, destination address, and time between packets. Countermeasures will be considered specifically for broadcast networks.

11.2.1 Packet Length

Closing the packet length channel is very simple. Require all packets to be the same fixed length, padding short packets out to the full length. Now, the length field is not needed by the communications processor, and the fact that the packet is padded should not be visible through the encryption. While padding all packets to maximum length certainly wastes some bandwidth for short packets, the single packet size can considerably simplify software buffer management. Therefore, the cost of fixed length packets might not be unreasonable.

11.2.2 Destination Address

Concealing the destination address of a packet is considerably more difficult than concealing the length. If the communications network must route packets based on address, then the address must appear in plaintext. (1) In a broadcast network, every packet, by definition, is transmitted to every host. (2) Therefore, each host could attempt to decrypt every incoming packet. If the host was not authorized to

---

(1) As a special case, if a host communicates with only one other host at one predetermined access class, then the destination address can be filled in as a constant by the security kernel or the cryptographic device, and therefore, "Trojan Horses" can no longer modulate the destination address. However, this limitation reduces the network to providing no more than the equivalent of a dedicated telephone line with link encryption.

(2) Examples of broadcast networks include the Xerox Ethernet <Metcalfe76>, the U. C. Irvine Distributed Computing System <Rowe75>, and the ALOHA network <Abramson70>.

receive the packet, the encryption keys would fail to match, and the plaintext produced by the decryption device would be garbage. If the packet successfully decrypted, then the host could check the now decrypted destination address to be sure it was his own, so that any packets for which the host was authorized, but were not addressed to that host could be discarded. For an n-bit host address, there is a probability of 1 in 2**n that a key mismatch will decrypt to the correct address. The probability of this type of error can be reduced arbitrarily by adding redundancy to the packet. Either a checksumming technique could be used, or the message stream authentication techniques described by Kent <Kent76> could be used.

11.2.3 Time Between Packets

Modulation of the time between packet transmissions is an example of Lampson's "covert channels" <Lampson73>. Lipner <Lipner75> points out that covert channels can be closed by making the observed time required for an event independent of the actual time. Therefore, if each host were assigned a fixed time slot for transmission, the time between packets would be guaranteed to always be equal. Fixed time slots can certainly exact a heavy penalty in bandwidth. The performance aspects of time slot allocation are examined in <Roberts73>.

11.3 Dynamic Key Renaming

In the previous section, the packet length channel was closed by using fixed length packets, and the time between packets channel was closed by using fixed time slots for transmission. However, the destination address channel was closed by requiring all hosts to attempt decryption of every incoming packet. Unfortunately, decrypting all messages is not as easy as it might seem. If a host is a multilevel host, servicing several hundred access classes, it may have to try, in the worst case, several hundred cryptographic keys to attempt to decrypt every incoming packet. (1) The host could try each key serially, but then could not keep up with the packet arrival rate. Alternatively, the host could try all the keys in parallel, but several hundred cryptographic devices connected in parallel would be far too unwieldy. Clearly, some other approach is needed to be able to select the correct key.

To solve the dilemma of serial versus parallel decryption, we borrow a technique from Farber and Larsen's Dynamic Process Renaming strategy <Farber75> that was previously discussed in section 5.4. Farber and Larsen's scheme suffered from the fact that the data in each packet was not encrypted. We propose here to encrypt the destination

(1) In the worst case, the multilevel host is communicating with several hundred dedicated hosts, all at different access classes. Therefore, each network connection would require a different cryptographic key. Such a situation could arise in a stock tranfer system, where "Trojan Horses" could be used to reveal the timing of large stock transactions, prior to their occuring.

address and data of each packet, and to use Farber and Larsen's renaming scheme, not to rename processes, but to rename cryptographic keys.

11.3.1 Name Generation

Assume that a name is associated with each cryptographic key in the system. The name could be a bit string long enough to be guaranteed

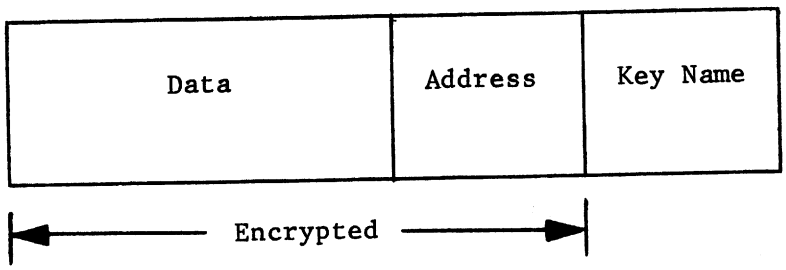


Figure 11.1 Typical Packet

unique over some long period of time. Now if every encrypted packet were preceded by its key name, as shown in figure 11.1, then the receiving host could easily select the correct cryptographic key looking up the key name in an associative memory. Because we have end-to-end encrypted the address field and data field, we are no longer vulnerable to the direct traffic analysis to which Farber and Larsen's scheme fell prey.

Of course, now we are back where we started with end-to-end encryption. A "Trojan Horse" could modulate key names just as well as destination addresses. However, if key names were changed after every packet, just as Farber and Larsen changed process names, then no external agent could discern a pattern in the use of key names. Figure 11.2 shows how a name generator could prefix every outgoing packet with a new name for its cryptographic key. Figure 11.3 shows how the name is stripped off an incoming packet and how the key is retrieved from the associative memory. After the key is retrieved, the next name is generated and stored in the associate memory.

Since the name generators must pick names at random without any predictability, the name generation function itself could be a high quality encryption algorithm. The example in the figures shows both the sender and receiver generating names. Alternatively, the sender could include the next name in the encrypted portion of the packet itself.

11.3.2 Sychronization

Similar to Farber and Larsen's scheme, dynamic key renaming has a synchronization problem. If a packet is lost, the name generator and the associative memory will never get back in synchrony. This problem can be reduced by generating several names in advance to be used if a packet seems to be lost. (1) These special resynchronization key names

(1) Presumably, timeouts are used to detect lost packets.

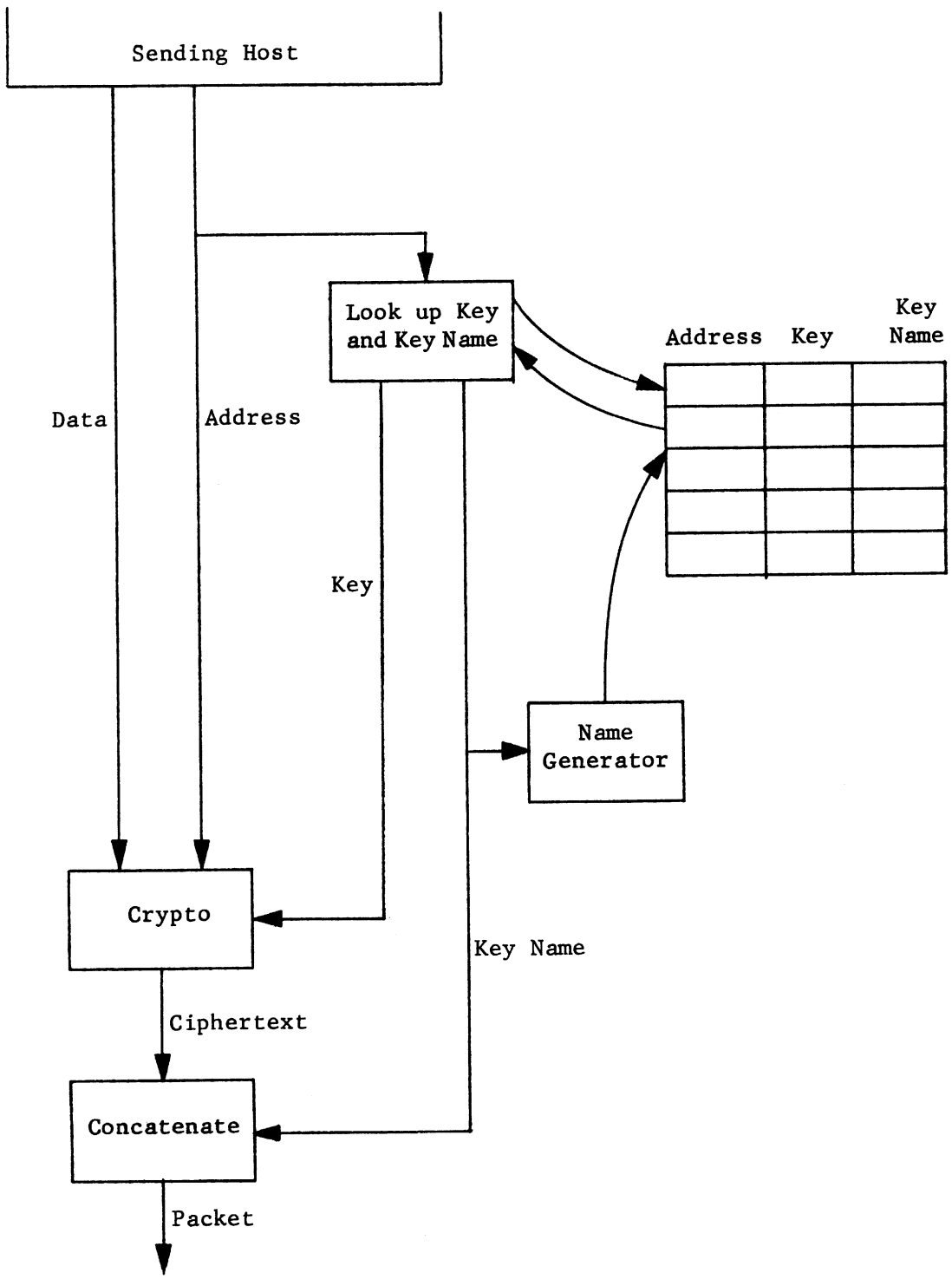


Figure 11.2 Transmission Name Generator

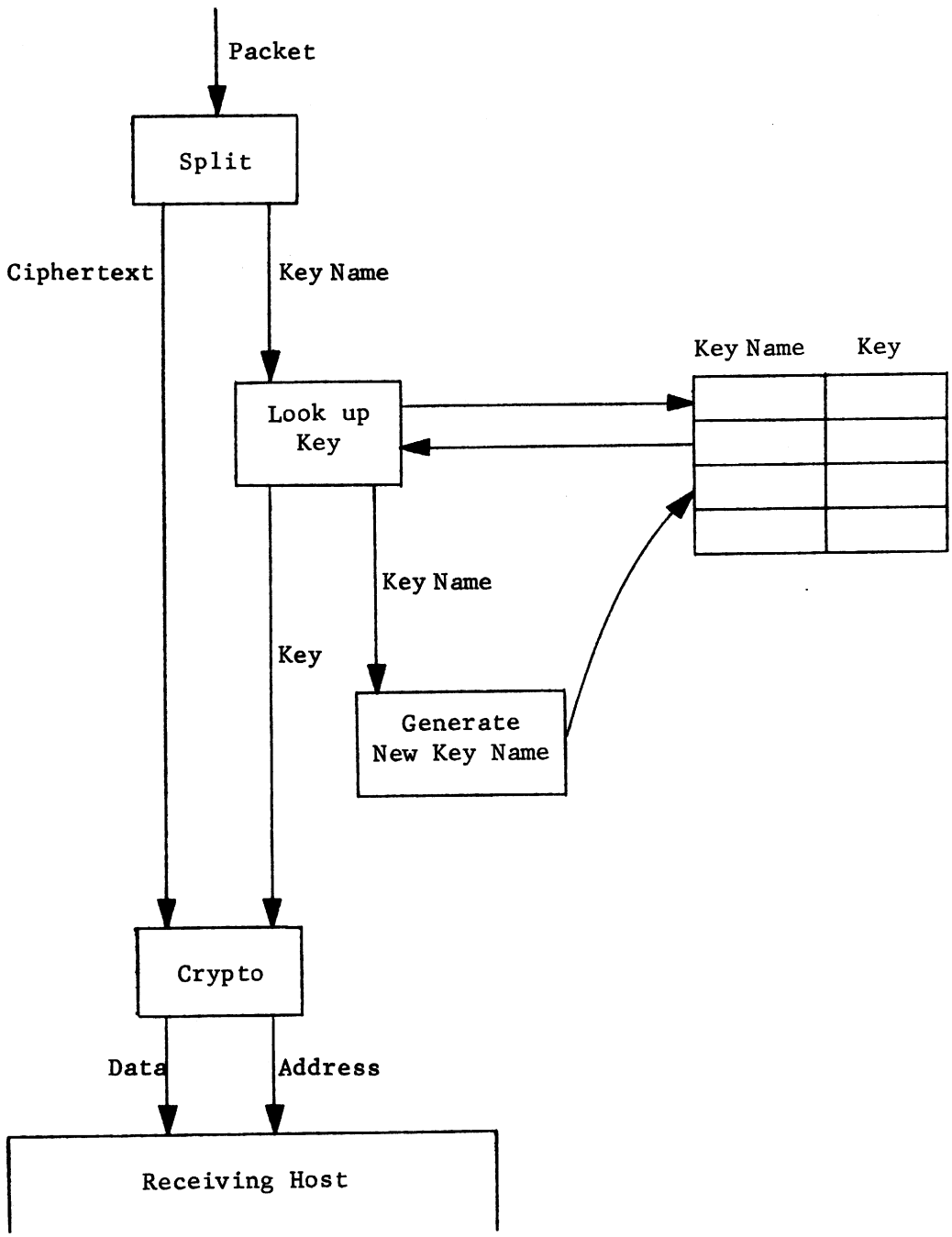


Figure 11.3 Reception Name Generator

could be used to recover from lost packets. If resynchronization also fails, then the network connection must be considered to be broken, and a new network connection must be established in order to continue.

11.3.3 Opening Connections

Network connections could be opened by requesting a cryptographic key and an initial key name from a Network Security Center (NSC) <Branstad75>. Communication with the NSC could also use dynamic key renaming, if the initial keys and key names were entered manually. Clearly, a large number of resynchronization key names should be reserved for use with the NSC, since loss of the NSC connection has severe consequences for a host.

Chapter Twelve

Authentication

Authentication becomes of interest in decentralized computing systems when we wish to forward an authentication from one host to another. If two hosts are to cooperate, some form of authentication must always be forwarded, although the forwarding may be implicit in some cases. In contrast to the rest of this thesis, both non-discretionary and discretionary aspects are covered, because of the high potential for confusion between authentication and access control. Forwarding of authentication between processes on a single host is described in <Montgomery76>. The schemes described below are extensions of his concepts to network systems.

12.1 Forwarding Authentication in the Lattice Model

Authentication forwarding is entirely implicit in the lattice model. As discussed in section 7.1.2, each packet or network port has an associated security label. From that label, any receiving process can immediately determine the access class of the sending process. The security labels are assured to be correct, because they are created either by the host security kernel if the sending host is multilevel, or by the network interface processor security kernel if the sending host is dedicated. As the labels are forwarded through the network, they are

protected from tampering by either encryption, if end-to-end encryption is used, or by the packet switch security kernels, if link encryption is used.

12.2 Forwarding Authentication in Discretionary Systems

In a discretionary security system, authentications cannot be passed implicitly like the security labels in the non-discretionary case. Users have names registered on various hosts of the network, and their access rights are determined as a function of those names. (1) The same user may have different names on different hosts. For example, a user may be named Smith on one host, JSmith on another, and M4409 on a third. However, all three names represent the same human being, and that human being would like to authenticate once and freely use all hosts on the network to which he or she has been granted access.

Sections 5.1.1, 5.1.2, and 5.1.3 describe two ways of forwarding authentications - storing passwords and trusting a central authority. Both these schemes have serious drawbacks. Stored passwords can be easily distributed to other users and may be compromised while passing through the network. In addition, when the user wishes to change a password, he or she must transmit the password to all other hosts that may have previously received it. While clearly not impossible, such

(1) Access is determined as a function of name in both Access Control List (ACL) systems and capability list systems. For example, the Cambridge Capability System <Slinn76> has the equivalent of an ACL to determine to whom a capability should be granted.

password distribution tasks can become quite onerous. The National Software Works (NSW) approach of establishing a central authority contradicts the basic goal of discretionary access controls - that of decentralized access decisions. The NSW's requirement that a host system implicitly trust the access decisions of a central authority may be totally unacceptable to the system administrator of some hosts. The system administrator may be willing to grant limited and well-bounded trust, but the NSW approach demands unlimited trust.

As an alternative to the stored passwords and the central authority, a scheme for forwarded authentication based on proxy login is proposed. Proxy login allows one user to grant a proxy to another user. When the second user wishes to exercise the proxy, he or she requests a proxy login from the system and gives his or her own password. Assuming the first user has given permission, the second user is logged into the system under the first user's ID. Proxy login has been proposed for Multics <Saltzer74>, but has never been implemented.

In a decentralized computing system, proxy login could be extended as follows. User Smith on host A grants a proxy to user JSmith on host B. Now, when JSmith on host B want to login to host A, JSmith sends a proxy login request to host A. Host B's operating system annotates the request with the information that the request came from JSmith on host B. Based on this information, host A can allow JSmith to login as Smith without presenting a password.

Because host B does not store a password for Smith at host A, user Smith knows that JSmith cannot give the password to anyone else, either deliberately or accidentally. Therefore, Smith need not broadcast his new host A password to many other hosts when it changes. Also, Smith can remove a name from the proxy list without having to change his password and therefore having to notify all other proxy recipients. It should be noted, however, that proxy login offers no advantage over stored passwords for protection from "Trojan Horses." User JSmith can still login as Smith, and any "Trojan Horse" that JSmith runs has full access to Smith's data, just as if it had Smith's password.

Proxy login also offers advantages to the system administrator (SA) of host A. The SA need not completely trust host B. The proxy allows host B to gain access only to Smith's data and no other. Assuming host A is basically secure, the proxy has limited the potential damage that host B can do. No such damage limiter is present in the central authority scheme.

Proxy login, however, does have several disadvantages. First, the system must provide a mechanism for defining proxies that is not vulnerable to "Trojan Horses." Presumably, a trusted process will meet this requirement.

Second, by granting proxy login to JSmith on host B, Smith has implicitly granted proxy login to any proxies that have been granted by JSmith. In addition, anyone who can penetrate the access controls of host B or the system administrator of host B (if he or she misuses his

or her powers) can now also proxy for Smith. While Smith should be aware of these potential problems, he may not think of them when granting a proxy. Thus, Smith may be "surprised" that he granted more access than he thought. This is a serious problem, because most security failures are caused by human errors, not hardware or software errors. The human engineering of the proxy login (and the entire security system) must be carefully planned to assure that users to not make inadvertent blunders that have disastrous consequences.

Proxy login has one other difficulty. User Smith will realize that he can also grant a proxy to his friend Jones on host C. Smith wants Jones to run one of his (Smith's) programs, but Jones does not have an account on host A. However, if Smith grants Jones a proxy, Jones now has access to all of Smith's data and all of Smith's money. (We assume that all computing resources must be paid for and that each user establishes an account against which computer usage is billed.) What Smith wants to do is limit Jones' access to just certain files and to a certain amount of money. One could invent ad hoc solutions to this problem, for example, using the Multics instance tag mechanism. However, a proper solution lies in implementing Rotenberg's authority hierarchy concept that would allow each level in an authority hierarchy to recursively define sub-authorities. Further investigation of this area is outside the scope of this thesis. The interested reader is referred to Rotenberg's PhD thesis <Rotenberg74>.

[This page intentionally left blank.]

Chapter Thirteen

Conclusions

13.1 Where Have We Been?

The goal of this thesis was to develop a consistent and effective approach to provide non-discretionary access controls for decentralized computing systems. To meet this goal, we first defined the semantics of the lattice model in light of expected threats. In particular, we defined the differences between dedicated hosts and multilevel hosts from the point of view of confinement. We then examined several levels of protocol for decentralized systems, linking each protocol to basic security requirements.

At the basic host to host protocol level, we showed how packets can be labelled with their access class to assure delivery only to authorized destinations. We outlined a mechanism for one-way communication between hosts dedicated to different access classes, and showed how limited forms of error and flow control could be supported.

At the first protocol level above basic packet communications, the lattice model was added to Reed's scheme for naming services in decentralized systems. We saw that access classes could be assigned to directories in the naming system, and that the compatibility requirement for non-decreasing access classes could be weakened considerably for

naming networks that allow multiple parents. However, we saw that revocation of access was essential to relaxing the compatibility requirements. If revocation were not possible, then relaxation of the compatibility requirements could leave objects in the naming network that could not be deleted.

We next examined the concept of downgrading information, and saw how decentralized processing can make downgrading easier to perform. The multilevel intelligent terminal was introduced to aid the human being who must perform downgrading operations. To support the multilevel terminal's security kernel, descriptor based display addressing was proposed to allow untrusted software direct access to windows of the display screen.

Decentralization of computing resources introduced new problems in administration of the lattice model. Access class proliferation, which caused some difficulty for Multics with only 18 categories, became a problem of the first magnitude when thousands of categories were contemplated in a national commercial network. A scheme was proposed to use Branstad's Network Security Centers to provide automated definition of new categories on demand. In addition, a scheme was proposed to assign unique numbers to categories and frequently used category sets. By using the unique numbers, a considerably more compact representation of access classes is possible.

Returning to the lowest level protocols again, we examined the use of end-to-end encryption in conjunction with the lattice model. Here, we found a fundamental limitation to the use of end-to-end encryption that results from the requirement that packet destination addresses appear in cleartext. We saw how this requirement was not present in broadcast networks, because all packets are broadcast to all possible receivers. However, encryption of the destination address led to the difficulty of identifying the correct decryption key from the potentially very large number of keys simultaneously in use by any given host. To resolve this problem, the strategy of dynamic key renaming was proposed.

Finally, we examined the area of authentication forwarding, both for non-discretionary and discretionary systems. For non-discretionary systems, authentication is implicitly forwarded from host to host in the access class labels on packets. However, for most discretionary systems, authentication forwarding is accomplished by transmitting passwords from host to host. As an alternative to transmitting passwords, we proposed a technique for forwarding authentication based on proxy login. This technique allows a user to accept forwarded authentications, without requiring unlimited trust of the foreign host system.

13.2 Where Can We Go?

Based on the ideas presented in this thesis, the pursuit of non-discretionary access control for decentralized systems should go in three major directions - implementation, legislation, and further research.

13.2.1 Implementation

Several of the concepts presented in this thesis can be implemented immediately. The protocols for host to host communication, both one-way and two-way, can be built into multilevel communications processors now. Such systems as SATIN IV and AUTODIN II will need these types of controls (and are planning them). Once networks with effective host to host security controls are operational, the naming schemes discussed in this thesis can be added to provide uniform secure naming of services across the entire networks. Experimental multilevel terminals can be built today using dedicated security kernel based processors. Until the costs are low enough to have on processor per terminal, a security kernel based processor could act as a concentrator for several displays, although this would significantly increase the complexity of the terminal controlling software and therefore, the difficulty of certification. One prototype multilevel terminal has already been developed <Ames76> using an 8080 microprocessor in an HP2649 display terminal, but this terminal has serious limitations due to its limited

screen capabilities, and due to its lack of a security kernel. A better multilevel terminal could be built using the Xerox SDWP with a security kernel. Finally, the proxy login concept could be easily added to systems presently on the ARPANET with little change in existing TELNET and FTP protocols.

13.2.2 Legislation

Throughout this thesis, we have assumed that categories provide an accurate model of the protection requirements for privacy. Based on a technical assessment of privacy, categories are indeed appropriate. However, current privacy legislation <Privacy74> is extremely vague concerning actual protection requirements. The present legislation requires only "adequate" protection of information, but leaves "adequate" undefined. If privacy is to be enforced by law, the semantics of protection of privacy must be clearly defined. Analogous to the Executive Order that defines the military classification system <Nixon72>, some type of legal definition of categories for privacy is needed. The definition should not list the precise categories to be used (since there may be many changing requirements for categories), but it should authorize the existence of categories, clearly define their semantics, and provide a legal mechanism for allocating categories.

13.2.3 Further Research

Further research in non-discretionary access control for decentralized systems can be profitably engaged in several areas. Most importantly, research is still required to develop security kernels for large general purpose systems. While this thesis has assumed the existence of large multilevel secure service hosts, development of such systems, as described in <Rhode77>, has been terminated prematurely. While one way communication protocols can provide a limited capability for dedicated systems, the full benefits of decentralization of computing can only be realized with multilevel secure hosts.

Second, as noted in section 8.2.5, research is needed in the area of garbage collection of multilevel, multihost data bases. Techniques are required to garbage collect, without violation of the confinement property.

Finally, further research is needed in the area of end-to-end encryption to find protocols that allow the use of end-to-end encryption in non-broadcast networks. End-to-end encryption offers a number of advantages that would make it desirable in non-broadcast networks, if the cleartext address problem could be resolved. (Alternatively, research to increase available communication bandwidths could eliminate the need for non-broadcast networks.)

REFERENCES

(Documents with NTIS numbers may be ordered from the National Technical Information Service, U.S. Department of Commerce, 5285 Port Royal Road, Springfield, Virginia 22151.)

<Abbott76> Abbott, R. P., et al., Security Analysis and Enhancements of Computer Operating Systems, The RISOS Project, Lawrence Livermore Laboratory, Livermore, Ca., NBSIR 76-1041, National Bureau of Standards, Washington, D.C., April 1976.

<Abramson70> Abramson, N., "The Aloha System," 1970 Fall Joint Computer Conference, AFIPS Conference Proceedings, Vol. 37, AFIPS Press, Montvale, N. J., 1970, pp. 281-285.

<Ames74> Ames, S. R., File Attributes and Their Relationship to Computer Security, S.M. thesis, Department of Computing and Information Sciences, Case Western Reserve University, Cleveland, Ohio. (Also available as ESD-TR-74-191, HQ Electronic Systems Division, Hanscom AFB, Ma., June 1974. (NTIS# AD A002159))

<Ames76> Ames, S. R., User Interface Multilevel Security Issues in a Transaction-Oriented Data Base Management System, MTP-178, The MITRE Corp., Bedford, Ma., December 1976.

<Anderson71> Anderson, James P., AF/ACS Computer Security Controls Study, James P. Anderson and Co., ESD-TR-71-395, HQ Electronic Systems Division, Hanscom AFB, Ma., November 1971. (NTIS# AD 251865L)

<Anderson72> Anderson, James P., Computer Security Technology Planning Study, James P. Anderson and Co., ESD-TR-73-51, Vols. I and II, HQ Electronic Systems Division, Hanscom AFB, Ma., October 1972. (NTIS# AD 758206 and NTIS# AD 772806)

<Attanasio76> Attansio, C. R., P. W. Markstein, and R. J. Phillips, "Penetrating an Operating System: A Study of VM/370 Integrity," IBM Systems Journal, Vol. 15, No. 1, 1976, pp. 102-116.

<Bell73> Bell, D. E. and L. J. LaPadula, Secure Computer Systems: A Mathematical Model, The MITRE Corp., ESD-TR-73-278, Vol. II, HQ Electronic Systems Division, Hanscom AFB, Ma., November 1973. (NTIS# AD 771543)

<Bell74> Bell, D. E., Secure Computing Systems: A Refinement of the Mathematical Model, The MITRE Corp., ESD-TR-73-278, Vol. III, HQ Electronic Systems Division, Hanscom AFB, Ma., April 1974. (NTIS# AD 780528)

<Bell75> Bell, D. E. and L. J. LaPadula, Computer Security Model: Unified Exposition and Multics Interpretation, The MITRE Corp., ESD-TR-75-306, HQ Electronic Systems Division, Hanscom AFB, Ma., June 1975. (NTIS# AD A023588)

<Biba77> Biba, K., G. Nibaldi, and J. Woodward, A Kernel-Based Secure UNIX Design, WP-21196, The MITRE Corp., Bedford, Ma., 6 April 1977.

<Branstad73> Branstad, D., "Security Aspects of Computer Networks," Proceedings of the AIAA Computer Network Systems Conference, Paper 73-427, Huntsville, Al., April 1973.

<Branstad75> Branstad, D., "Encryption Protection in Computer Data Communications," Fourth Data Communications Symposium, Quebec City, Canada, 7-9 October 1975, pp. 8-1 - 8-7.

<Bressler76> Bressler, R. D., Combined Quarterly Technical Report No. 3: Private Line Interface Development, Packet Network Security, Packet Broadcast by Satellite, BBN Report No. 3458, Bolt, Beranek and Newman, Inc., Cambridge, Ma., November 1976. (NTIS# AD A033352)

<Broadbridge76> Broadbridge, R. and J. Mekota, Secure Communications Processor Specifications, Honeywell Information Systems, Inc., ESD-TR-76-351, Vol. II, HQ Electronic Systems Division, Hanscom AFB, Ma., June 1976.

<Burke74> Burke, E. L., Concept of Operations for Handling I/O in a Secure Computer at the Air Force Data Services Center (AFDSC), The MITRE Corp., ESD-TR-74-113, HQ Electronic Systems Division, Hanscom AFB, Ma., April 1974. (NTIS# AD 780520)

<Cave Brown75> Cave Brown, A., Bodyguard of Lies, Harper & Row, New York, N. Y., 1975.

<Cerf74> Cerf, V. G. and R. E. Kahn, "A Protocol for Packet Network Intercommunications," IEEE Transactions on Communications, Vol. COM-22, No. 5, May 1974, pp. 637-648.

<Chandersekaran76> Chandersekaran, C. S. and K. S. Shankar, "Towards Formally Specifying Communication Switches," Trends and Applications 1976: Computer Networks, Institute of Electrical and Electronics Engineers, Inc., New York, N. Y., 17 November 1976, pp. 104-112.

<Cohen75> Cohen, E. and D. Jefferson, "Protection in the Hydra Operating System," Proceedings of the Fifth Symposium on Operating System Principles, ACM Operating Systems Review, Vol. 9, No. 5, November 1975, pp. 141-160.

<Computerworld75> "T/S Service Security Cracked by Schoolboy With Series of Tricks," Computerworld, Vol. IX, No. 5, 29 January 1975.

<Cotton75> Cotton, I. W. and P. Meissner, "Approaches to Controlling Personal Access to Computer Terminals," Proceedings of the 1975 Symposium on Computer Networks: Trends and Applications, Institute of Electrical and Electronics Engineers, New York, N. Y., 1975.

<Danet76> Danet, A., et al., "The French Public Packet Service: The Transpac Network," Proceedings of the Third International Conference on Computer Communication, Toronto, Canada, 3-6 August 1976, pp. 251-260.

<Denning76> Denning, D. E., "A Lattice Model of Secure Information Flow," Communications of the ACM, Vol. 19, No. 5, May 1976, pp. 236-243.

<Diffie76> Diffie, W. and M. E. Hellman, "Multiuser Cryptographic Techniques," 1976 National Computer Conference, AFIPS Conference Proceedings, Vol. 45, AFIPS Press, Montvale, N. J., 1976, pp. 109-112.

<Dijkstra68> Dijkstra, E. W., "Cooperating Sequential Processes," Programming Languages (Ed. F. Genuys), Academic Press, New York, N. Y., 1968.

<DoD73> ---, ADP Security Manual: Techniques and Procedures for Implementing, Deactivating, Testing, and, Evaluating Secure Resource-Sharing ADP Systems, DoD 5200.28-M, Department of Defense, Washington, D.C., January 1973.

<Engelbart68> Engelbart, D. C. and W. K. English, "A Research Center for Augmenting Human Intellect,", 1968 Fall Joint Computer Conference, AFIPS Conference Proceedings, Vol. 33, Part 1, AFIPS Press, Montvale, N. J., 1968, pp. 395-410.

<ESD74> ---, The Feasibility of a Secure Communications Executive for a Communications System, MCI-75-10, Information Systems Technology Applications Office, HQ Electronic Systems Division, Hanscom AFB, Ma., August 1974.

<Everett57> Everett, R. R., C. A. Zraket, and H. D. Bennington, "SAGE - A Data-Processing System for Air Defense," Proceedings of the Eastern Joint Computer Conference, December 9-13, 1957, Washington, D. C., The Institute of Radio Engineers, Inc., New York, N. Y., 1958.

<Fabry74> Fabry, R. S., "Capability-Based Addressing," Communications of the ACM, Vol. 17, No. 7, July 1974, pp. 403-412.

<Farber75> Farber, D. J. and K. C. Larson, "Network Security Via Dynamic Process Renaming," Fourth Data Communications Symposium, Quebec City, Canada, 7-9 October 1975, pp. 8-13 - 8-18.

<Feinler76> Feinler, E. and J. Postel, ARPANET Protocol Handbook, NIC 7104, Network Information Center, Stanford Research Institute, Menlo Park, Ca., April 1976.

<Gardella76> Gardella, R. S., Issues in the Design and Use of Secure Terminals, MTR-3128, The MITRE Corp., Bedford, Ma., June 1976.

<Harrison76> Harrison, M. A., W. L. Ruzzo, and J. D. Ullman, "Protection in Operating Systems," Communications of the ACM, Vol. 19, No. 8, August 1976, pp. 461-471.

<Hartke77> Hartke, D., W. Sterling, and J. Shemer, "A Raster Scan CRT Display Processor for Typewriter Emulation," Fourteenth IEEE Computer Society International Conference, IEEE, Inc., New York, 1977, pp. 54-58.

<Heinrich76> Heinrich, F. R. and D. J. Kaufman, "A Centralized Approach to Computer Network Security," 1976 National Computer Conference, AFIPS Conference Proceedings, Vol. 45, AFIPS Press, Montvale, N. J., June 1976, pp. 85-90.

<Hennie77> Hennie, F., Introduction to Computability, Addison-Wesley, Reading, Ma., 1977.

<Hennigan76> Hennigan, K. B., Hardware Subverter for the Honeywell 6180, The MITRE Corp., ESD-TR-76-352, HQ Electronic Systems Division, Hanscom AFB, Ma., December 1976.

<Hoffman70> Hoffman, L. J., The Formulary Model for Access Control and Privacy in Computer Systems, PhD thesis, Department of Computer Science, Stanford University, May 1970. (Also available as Stanford Linear Accelerator Center Report No. 117, Stanford University, Stanford, Ca., May 1970.)

<Honeywell76> ---, Security Kernel Specification for a Secure Communications Processor, Honeywell Information Systems, Inc., ESD-TR-76-359, HQ Electronic Systems Division, Hanscom AFB, Ma., in preparation.

<IMP76> ---, Specifications for the Interconnection of a Host and an IMP, Report No. 1822, Bolt Beranek and Newman, Inc., January 1976, Appendix H.

<Kahn67> Kahn, D., The Codebreakers, Macmillan, New York, N. Y., 1967.

<Kampe77> Kampe, M. and G. Popek, The UCLA Data Secure UNIX Operating System, University of California at Los Angeles, 1977.

<Karger74> Karger, P. A. and R. R. Schell, Multics Security Evaluation: Vulnerability Analysis, ESD-TR-74-193, Vol. II, HQ Electronic Systems Division, Hanscom AFB, Ma., June 1974. (NTIS# AD A001120)

<Kent76> Kent, S. T., Encryption - Based Protection Protocols for Interactive User - Computer Communication, SM thesis, M.I.T. Dept. of Electrical Engineering and Computer Science, May 1976. (Also available as MIT/LCS/TR-162, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, Ma., May 1976. (NTIS# AD A026911))

<Lampson71> Lampson, B. W., "Protection," Proceedings Fifth Princeton Conference on Information Sciences and Systems, Princeton University, March 1971, pp. 437-443, reprinted in Operating Systems Review, Vol. 8, No. 1, January 1974, pp. 18-24.

<Lampson73> Lampson, B. W., "A Note on the Confinement Problem," Communications of the ACM, Vol. 16, No. 10, October 1973, pp. 613-615.

<Lampson76> Lampson, B. W. and H. E. Sturgis, "Reflections on an Operating System Design," Communications of the ACM, Vol. 19, No. 5, May 1976, pp. 251-265.

<Lampson77> Lampson, B. W., et al., "Report on the Programming Language Euclid," ACM SIGPLAN Notices, Vol. 12, No. 2, February 1977.

<Lipner71> Lipner, S. B., MACIMS Security Configurations, WP-3697, The MITRE Corp., Bedford, Ma., 6 January 1971.

<Lipner72> Lipner, S. B., SATIN Computer Security, The MITRE Corp., MCI-75-2, Information Systems Technology Applications Office, HQ Electronic Systems Division, Hanscom AFB, Ma., September 1972.

<Lipner75> Lipner, S. B., "A Comment on the Confinement Problem," Proceedings of the Fifth Symposium on Operating System Principles, ACM Operating Systems Review, Vol. 9, No. 5, November 1975, pp. 192-196.

<Liskov77> Liskov, B, et al., "Abstraction Mechanisms in CLU," to appear in Communications of the ACM, Vol. 20, No. 7, July 1977.

<Mack76> Mack, J. L. and B. N. Wagner, "Secure Multilevel Data Base System: Demonstration Scenarios," The MITRE Corp., ESD-TR-76-158, HQ Electronic Systems Division, Hanscom AFB, Ma., October 1976.

<MACSYMA75> MACSYMA Reference Manual, Project MAC, Massachusetts Institute of Technology, Cambridge, Ma., November 1975.

<MAM76> ---, Multics Administrators' Manual - System Administrator, AK50, Rev. 1, Honeywell Information Systems, Inc., Waltham, Ma., October 1976.

<Metcalfe76> Metcalfe, R. M. and D. R. Boggs, "Ethernet: Distributed Packet Switching for Local Computer Networks," Communications of the ACM, Vol. 19, No. 7, July 1976, pp. 395-404.

<Millen76> Millen, J. K., "Security Kernel Validation in Practice," Communications of the ACM, Vol. 19, No. 5, May 1976, pp. 243-250.

<Millstein76> Millstein, R., National Software Works Status Report No. 1, Massachusetts Computer Associates, RADC-TR-76-276, Vol. I, Rome Air Development Center, Griffiss AFB, N. Y., September 1976. (NTIS# AD A034133)

<Montgomery76> Montgomery, W. A., A Secure and Flexible Model of Process Initiation for a Computer Utility, SM and EE thesis, MIT Dept. of Electrical Engineering and Computer Science, June 1976. (Also available as TR-163, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, Ma., June 1976.)

<MPM75> ---, Multics Programmers' Manual Reference Guide, AG91, Rev. 1, Honeywell Information Systems, Inc., Waltham, Ma., December 1975.

<Nibaldi76> Nibaldi, G. and B. Wagner, Secure Multilevel Data Base Systems: System Software, MTR-3160, Vol. III, The MITRE Corp., Bedford, Ma., in preparation.

<Nixon72> Nixon, R. M., "Classification and Declassification of National Security Information and Material," Executive Order 11652, The White House, Washington, D. C., 8 March 1972.

<Organick72> Organick, E. I., The Multics System: An Examination of its Structure, MIT Press, Cambridge, Ma., 1972.

<Padlipsky77> Padlipsky, M., D. Snow, and P. Karger, Limitations of End-to-End Encryption in Secure Communications Networks, MTR-XXXX, The MITRE Corp., Bedford, Ma., in preparation.

<Painter75> Painter, J. A., "A Minicomputer Network to Enhance Computer Security," Communications Networks, Online Conferences, Ltd., Uxbridge, England, 1975.

<Parker73> Parker, D. B., S. Nycum, and S. S. Oüra, Computer Abuse, Stanford Research Institute, Menlo Park, Ca., November 1973.

<Popek74> Popek, G. J. and C. S. Kline, "Verifiable Secure Operating System Software," 1974 National Computer Conference, AFIPS Conference Proceedings, Vol. 43, AFIPS Press, Montvale, N. J., 1974, pp. 135-142.

<Postel76> Postel, J. B., L. L. Garlick, and R. Rom, Transmission Control Protocol Specification, SRI-ARC-35938, SRI-ARC-35939, Augmentation Research Center, Stanford Research Institute, Menlo Park, Ca., 15 July 1976. (NTIS# AD A035337)

<Pouzin76> Pouzin, L., "Virtual Circuits vs. Datagrams - Technical and Political Problems," 1976 National Computer Conference, AFIPS Conference Proceedings, Vol. 45, AFIPS Press, Montvale, N. J., 1976, pp. 483-494.

<Privacy74> Privacy Act of 1974, Title 5, United States Code, Section 552a (Public Law 93-579), December 31, 1974.

<Redell74> Redell, D. D., Naming and Protection in Extendible Operating Systems, PhD thesis, Department of Electrical Engineering and Computer Science, University of California at Berkeley. (Also available as Project MAC TR-140, Massachusetts Institute of Technology, Cambridge, Ma., November 1974. (NTIS# AD A001721))

<Reed77> Reed, D. P. and R. J. Kanodia, "Synchronization with Eventcounts and Sequencers," submitted to the Sixth ACM Symposium on Operating Systems Principles, Massachusetts Institute of Technology, Cambridge, Ma., March 29, 1977.

<Rhode77> Rhode, R. D., ESD 1976 Computer Security Developments Summary, The MITRE Corp., MCI-76-2, Directorate of Computer Systems Engineering, HQ Electronic Systems Division, Hanscom AFB, Ma., January 1977.

<Richardson73> Richardson, M. H. and J. V. Potter, Design of a Magnetic Card Modifiable Credential System Demonstration, MCI-73-3, HQ Electronic Systems Division, Hanscom AFB, Ma., December 1973.

<Ritchie74> Ritchie, D. M. and K. Thompson, "The UNIX Time-Sharing System," Communications of the ACM, Vol. 17, No. 7, July 1974, pp. 365-375.

<Roberts73> Roberts, L. G., "Dynamic Allocation of Satellite Capacity Through Packet Reservation," 1973 National Computer Conference and Exposition, AFIPS Conference Proceedings, Vol. 42, AFIPS Press, Montvale, N. J., 1973, pp. 711-716.

<Rotenberg74> Rotenberg, L. J., Making Computers Keep Secrets, PhD Thesis, Department of Electrical Engineering, Massachusetts Institute of Technology, February 1974. (Also available as Project MAC TR-115, Massachusetts Institute of Technology, Cambridge, Ma., February 1974. (NTIS# PB 229352/AS))

<Rowe75> Rowe, L. A., The Distributed Computing System, Technical Report #66, University of California at Irvine, Irvine, Ca., June 1975.

<Saltzer74> Saltzer, J., "Protection and the Control of Information Sharing in Multics," Communications of the ACM, Vol. 17, No. 7, July 1974, pp. 388-402.

<Schacht76> Schacht, J. M., Jobstream Separator: Supportive Information, The MITRE Corp., ESD-TR-75-354, HQ Electronic Systems Division, Hanscom AFB, Ma., January 1976. (NTIS# AD A020521)

<Schell73> Schell, R. R., P. J. Downey, and G. J. Popek, Preliminary Notes on the Design of Secure Military Computer Systems, MCI-73-1, HQ Electronic Systems Division, Hanscom AFB, Ma., January 1973.

<Schell76> Schell, R. R. and P. A. Karger, Security in Automatic Data Processing (ADP) Network Systems, ESD-TR-77-19, HQ Electronic Systems Division, Hanscom AFB, Ma., December 1976.

<Schiller73> Schiller, W. L., Design of a Security Kernel for the PDP-11/45, The MITRE Corp., ESD-TR-73-294, HQ Electronic Systems Division, Hanscom AFB, Ma., June 1973. (NTIS# AD 772808)

<Schiller75> Schiller, W. L., The Design and Specification of a Security Kernel for the PDP-11/45, The MITRE Corp., ESD-TR-75-69, HQ Electronic Systems Division, Hanscom AFB, Ma., May 1975. (NTIS# AD A011712)

<Schiller76> Schiller, W. L., P. T. Withington, and J. P. L. Woodward, Top Level Specification of a Multics Security Kernel, WP-20810, The MITRE Corp., Bedford, Ma., 9 July 1976.

<Schroeder75> Schroeder, M. D., "Engineering a Security Kernel for Multics," Proceedings of the Fifth Symposium on Operating System Principles, ACM Operating Systems Review, Vol. 9, No. 5, November 1975, pp. 25-32.

<Shaw77> Shaw, M. and W. A. Wulf, "Abstraction and Verification in ALPHARD: Defining and Specifying Iteration and Generators," to appear in Communications of the ACM, Vol. 20, No. 7, July 1977.

<Shemer77> Shemer, J. E., et al, "Development of an Experimental Display Word Processor for Office Applications," Fourteenth IEEE Computer Society International Conference, IEEE, Inc., New York, 1977, pp. 42-46.

<Slinn76> Slinn, C. J., Chaos Manual, Computer Laboratory, Corn Exchange Street, Cambridge, England, 3 September 1976.

<Stork75> Stork, D. F., Downgrading in a Secure Multilevel Computer System: The Formulary Concept, The MITRE Corp., ESD-TR-75-62, HQ Electronic Systems Division, Hanscom AFB, Ma., May 1975. (NTIS# AD A011696)

<Thomas73> Thomas, R. H., "A Resource Sharing Executive for the ARPANET," 1973 National Computer Conference and Exposition, AFIPS Conference Proceedings, Vol. 42, AFIPS Press, Montvale, N. J., 1973, pp. 155-163.

<Turn76> Turn, R., "Classification of Personal Information for Privacy Protection Purposes," 1976 National Computer Conference, AFIPS Conference Proceedings, Vol. 45, AFIPS Press, Montvale, N. J., 1976, pp. 301-307.

<Walter74> Walter, K. G., et al, Primitive Models for Computer Security, Case Western Reserve University, ESD-TR-74-117, HQ Electronic Systems Division, Hanscom AFB, Ma., 23 January 1974. (NTIS# AD 778467)

<Walter75> Walter, K. G., et al., Initial Structured Specifications for an Uncompromisable Computer Security System, Case Western Reserve University, ESD-TR-75-82, HQ Electronic Systems Division, Hanscom AFB, Ma., July 1975.

<Weissman69> Weissman, C., "Security Controls in the ADEPT-50 Time Sharing System," 1969 Fall Joint Computer Conference, AFIPS Conference Proceedings, Vol. 35, AFIPS Press, Montvale, N. J., 1969, pp. 119-133.

<White75> White, J. C. C., Design of a Secure File Management System, The MITRE Corp., ESD-TR-75-57, HQ Electronic Systems Division, Hanscom AFB, Ma., April 1975. (NTIS# AD A010590)

<Whitmore73> Whitmore, J., et al, Design for Multics Security Enhancements, Honeywell Information Systems, Inc., ESD-TR-74-176, HQ Electronic Systems Division, Hanscom AFB, Ma., December 1973. (NTIS# AD A030801)

Scanning Agent Identification Target

Scanning of this document was supported in part by the **Corporation for National Research Initiatives,** using funds from the **Advanced Research Projects Agency** of the **United states Government** under Grant: **MDA972-92-J1029.**

The scanning agent for this project was the **Document Services** department of the **M.I.T Libraries.** Technical support for this project was also provided by the **M.I.T. Laboratory for Computer Sciences.**

Scanned

Date: 11/30/1995

M.I.T. Libraries
Document Services

CS-TR Scanning Project
Document Control Form

Date : 11/3/95

Report # LCS-TR-179

Each of the following should be identified by a checkmark:
Originating Department:

☐ Artificial Intellegence Laboratory (AI)
☒ Laboratory for Computer Science (LCS)

Document Type:

☒ Technical Report (TR) ☐ Technical Memo (TM)
☐ Other:_____

Document Information

Number of pages: 140 (146-IMAGES)
Not to include DOD forms, printer intstructions, etc... original pages only.

Originals are:
☐ Single-sided or
☒ Double-sided

Intended to be printed as :
☐ Single-sided or
☒ Double-sided

Print type:
☒ Typewriter ☐ Offset Press ☐ Laser Print
☐ InkJet Printer ☐ Unknown ☐ Other:_____

Check each if included with document:

☐ DOD Form ☐ Funding Agent Form ☒ Cover Page
☒ Spine ☐ Printers Notes ☐ Photo negatives
☐ Other: _____

Page Data:

Blank Pages(by page number):_____

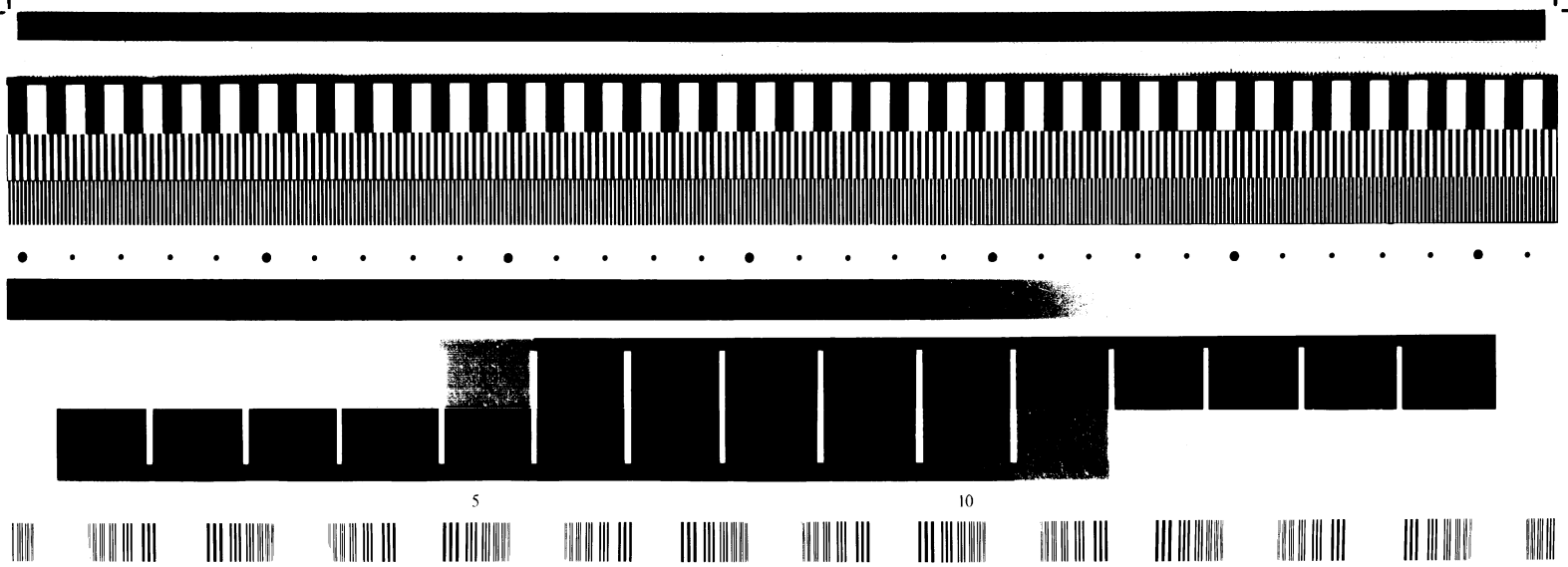
Photographs/Tonal Material (by page number):_____

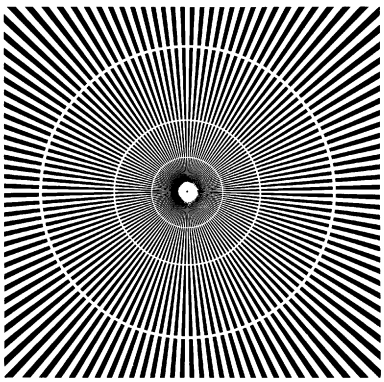
Other (note description/page number):

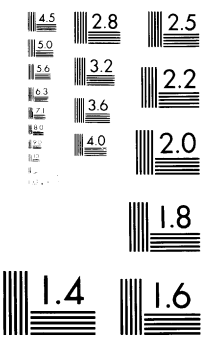
Description : Page Number:
IMAGE MAP: (1-140) UNN'ED TITLE PAGE, 2-139, UNN'ED BLANK
(141-146) SCANCONTROL, COVER, SPINE, TRGT'S (3)

Scanning Agent Signoff:
Date Received: 11/3/95 Date Scanned: 11/30/95 Date Returned: 12/7/95

Scanning Agent Signature: Michael W. Cook





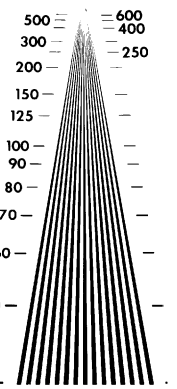


NMA MICROFONT QJKLPYZ
6BS12GH5D4X7U3W8V9E
PQR45DE9UV670FG8STHIJNOWXABYZ
3KLM12C

ABCDEFGHIJKLMNOPQRSTUVW
XYZabcdefghijklmnopqrst
uvwxyz0123456789 OCR-B

ABCDEFGHIJKLMNOPQRSTUV
WXYZabcdefghijklmnopqr
stuvwxyz1234567890PICA

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
1234567890 Elite



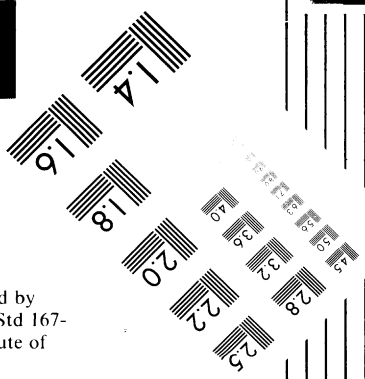
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
1234567890 Spartan Medium 6 pt

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
1234567890 Spartan Medium 4 pt

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
1234567890 Spartan Medium 8 pt

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
1234567890 Spartan Medium 10 pt

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
1234567890 Spartan Medium 12 pt



IEEE Std 167A-1987

FACSIMILE TEST CHART

Prepared by the IEEE Facsimile Subcommittee and printed by Eastman Kodak Company. Use in accordance with IEEE Std 167-1966, Test Procedure for Facsimile. Copyright 1987, Institute of Electrical and Electronics Engineers.

AIIM SCANNER TEST CHART # 2

Spectra

4 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789
6 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789
8 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789
10 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789

Times Roman

4 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789
6 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789
8 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789
10 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789

Century Schoolbook Bold

4 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789
6 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789
8 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789
10 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789

News Gothic Bold Reversed

4 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789
6 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789
8 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789
10 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789

Bodoni Italic

4 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789
6 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789
8 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789
10 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789

Greek and Math Symbols

4 PT ΑΒΓΔΕΞΘΗΙΚΛΜΝΟΠΦΡΣΤΥΩΧΨΖαβγδεξθηικλμνοπφρστυωχψζ≧∓",./≦±=≠°><⋊⋉≤≡
6 PT ΑΒΓΔΕΞΘΗΙΚΛΜΝΟΠΦΡΣΤΥΩΧΨΖαβγδεξθηικλμνοπφρστυωχψζ≧∓",./≦±=≠°><⋊⋉≤≡
8 PT ΑΒΓΔΕΞΘΗΙΚΛΜΝΟΠΦΡΣΤΥΩΧΨΖαβγδεξθηικλμνοπφρστυωχψζ≧∓",./≦±=≠°><⋊⋉≤≡
10 PT ΑΒΓΔΕΞΘΗΙΚΛΜΝΟΠΦΡΣΤΥΩΧΨΖαβγδεξθηικλμνοπφρστυωχψζ≧∓",./≦±=≠°><⋊⋉≤≡

White

Black

Isolated Characters

e	m	1	2	3	a
4	5	6	7	o	°
8	9	0	h	l	B

MESH HALFTONE WEDGES

65
85
100
110
133
150

A4 Page 6543210