ESD-TR-74-176

DESIGN FOR MULTICS SECURITY ENHANCEMENTS

J. Whitmore
A. Bensoussan
P. Green
D. Hunt
A. Kobziar
J. Stern

Honeywell Information Systems, Inc.
575 Technology Square
Cambridge, MA 02139

December 1973

Prepared for

DEPUTY FOR COMMAND AND MANAGEMENT SYSTEMS
ELECTRONIC SYSTEMS DIVISION
HANSCOM AIR FORCE BASE, MA 01731

LEGAL NOTICE

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

OTHER NOTICES

Do not return this copy. Retain or destroy.

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

F. WAH LEONG, CAPT, USAF
Project Officer
Air Force Data Services Center

ROGER R. SCHELL, MAJOR, USAF
Project Engineer

FOR THE COMMANDER

FRANK J. EMMA, COLONEL, USAF
Director, Information Systems
Technology Application Office
Deputy for Command & Management Systems

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: ESD-TR-74-176
2. GOVT ACCESSION NO.:
3. RECIPIENT'S CATALOG NUMBER:

4. TITLE *(and Subtitle)*: DESIGN FOR MULTICS SECURITY ENHANCEMENTS
5. TYPE OF REPORT & PERIOD COVERED:
6. PERFORMING ORG. REPORT NUMBER:

7. AUTHOR(s): J. Whitmore, A. Bensoussan, P. Green, D. Hunt, A. Kobziar, J. Stern
8. CONTRACT OR GRANT NUMBER(s): F19628-73-D-0087

9. PERFORMING ORGANIZATION NAME AND ADDRESS: Honeywell Information Systems, Inc. 575 Technology Square Cambridge, MA 02139
10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: Task 0004AA

11. CONTROLLING OFFICE NAME AND ADDRESS: Deputy for Command and Management Systems Electronic Systems Division Hanscom Air Force Base, MA 01731
12. REPORT DATE: December 1973
13. NUMBER OF PAGES: 93

14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*:
15. SECURITY CLASS. *(of this report)*: UNCLASSIFIED
15a. DECLASSIFICATION/DOWNGRADING SCHEDULE: N/A

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited.

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

18. SUPPLEMENTARY NOTES

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

computer security
security
access control
Multics
containment
operating system

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

The results of a 1973 security study of the Multics computer system are presented detailing requirements for a new access control mechanism that would allow two levels of classified data to be used simultaneously on a single Multics system. The access control policy was derived from the Department of Defense Information Security Program. The design decisions presented were the basis for subsequent security enhancements to the Multics system.

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

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

Contents

Contents (continued)

Preface

This report documents the results of a 1973 study to identify a set of security enhancements for Honeywell's Multics operating system. These enhancements were derived from the Department of Defense Information Security Program. The purpose of these enhancements was to permit users of two different security levels to simultaneously access classified information stored on the Multics system at the Air Force Data Services Center (AFDSC). This report served as a design document for the subsequent implementation of the security enhancements for use at the AFDSC.

The implementation of the design was based upon the "non-malicious" user concept. This concept is predicated upon the assumption that none of the user population would attempt malicious, concerted efforts to circumvent the enhanced security controls. The issues of guaranteeing the impenetrability of the security enhancements were not completely addressed, and the report makes no claim to the system's impenetrability. However, the proposed security controls are thought to be representative of those controls which could be provided on a certifiably secure system. The issues involved in the development of a certifiably secure system are the subject of a separate effort sponsored by the Information Systems Technology Applications Office of the Air Force's Electronic Systems Division.

During the course of the implementation of the security enhancements proposed in this report, several minor design changes were made. This report has not been updated to reflect these changes. This report should be taken neither as a precise description of the enhanced Multics system implemented for AFDSC nor as a description of Honeywell's Multics product--current or future.

INTRODUCTION

Honeywell participated in a joint Security Design Analysis with the Air Force to evaluate the requirements for providing a two-level security system on Multics. The primary goal was to develop a high level design for modifications to the Multics system to support a two-level security environment. This effort is a first step on the path to a certified secure system.

The analysis was conducted by a team composed of representatives from groups active in the computer security field. Team members were:

USAF AFDSC	Capt. F. Wah Leong Capt. Dave Schafer
USAF ESD	Major Roger Schell Lt. Paul Karger
MITRE Corp.	Steven Lipner Morrie Gasser Edmund Burke
Honeywell DSD	Jerold Whitmore Paul Green Douglas Hunt Jerry Stern
Honeywell CISL	Andre Bensoussan Andrew Kobziar

The Security Design Analysis covered the period from 10 July 1973 through 8 October 1973. The minutes of the weekly meetings are not part of this report.

This report was written by Honeywell personnel with review and guidance from the other team members. Responsibility for errors and omissions remains strictly with Honeywell.

Suggestions and design decisions contained in this report are not binding on either the Air Force or on Honeywell.

1. SCOPE OF THE SECURITY DESIGN ANALYSIS

1.1 Identification and Authority

The authority for this Security Design Analysis is contained in contract number F19628-73-0-0087. The Design Analysis task has been conducted as a joint effort of Honeywell Information Systems Inc., Data Systems Operations; Air Force Data Services Center; Air Force Electronics Systems Division (MCIT); and MITRE Corporation.

1.2 Purpose

1.2.1 Task Description

The primary task is to examine the problems and implications of operating the Honeywell Multics System in a restricted multi-level security mode for Secret and Top Secret cleared users. The primary criterion to be used in evaluating solutions to various problems is that the system should provide reasonable assurance that no Top Secret information can be compromised to a Secret cleared person. This means that on a single Multics system, within design constraints, there should be no information paths between users having different clearances which do not exist between users of physically separate dedicated computer systems.

With these problems in mind, the team looked for modifications to the Multics Operating System which will correct these problems, insofar as possible, and yet maintain the current user interface and functional capabilities of Multics. Specific design goals included:

1. Design to the requirements of the Air Force Data Services center RFP No: F19628-73-R-0024.

2. Design the basic security controls as an integral part of the Multics system.

3. Provide a design which may be extended for additional security enhancements.

4. Provide a generalized design that may be adapted for other DoD and commercial applications of the security system.

1.2.2 Specific Exclusions from the Design Analysis

Certain problems of multi-level security ADP operation and extensions of basic multi-level security controls were known at the start of the Design Analysis and were specifically excluded. These are described in the following paragraphs.

1.2.2.1 Certification

The task of certifying the correctness of any implementation of the multi-level security system design proposed in this report is, of course, beyond the scope of the Design Analysis. No hardware modifications are in fact required. In spite of a conceptually correct design, an actual implementation could conceivably contain programming errors which cause the system to behave incorrectly. Hence, absolute security cannot be claimed without certification. Consequently, in choosing among design alternatives, consideration has been given to facilitating the future task of certification.

1.2.2.2 The Trojan Horse Problem

A computer system which provides sharing of user written procedures is susceptible to a "Trojan Horse attack" by a malicious user. A Trojan Horse is a procedure which provides a potentially useful function to attract use by a person having access privileges not possessed by the author of the procedure. The Trojan Horse program detects such use and performs unauthorized or unwanted functions which would allow the author of the procedure to obtain information to which he did not otherwise have access or to perform acts of sabotage which would not otherwise be possible.

A general solution to the Trojan Horse problem is excluded from the scope of the Design Analysis. However, reducing the information paths between users of different clearance levels is within the scope of the Design Analysis. The issue of sabotage from a Trojan Horse is accepted with a low expectation of occurrence since all users of the system will

be cleared and assumed trustworthy. An act of sabotage at the AFDSC installation will have considerably less severe consequences than at certain other military sites such as those having a command and control environment.

1.2.2.3 High-Water Mark

The design extension of having users start work at a low level with automatic or requested upgrade to a higher level as more sensitive data is needed was specifically excluded from the scope of the Design Analysis. This extension is commonly described as a "high-water mark" capability.

1.2.2.4 Program Trustworthiness

The ability to reduce the system recognized clearance of a user who may attempt to access sensitive material, based on the clearance level of procedures executed in a user's process, is commonly described as the "trustworthiness" capability. This is one means to reduce the potential damage by a Trojan Horse attempting to perform sabotage. The "trustworthiness" capability is specifically excluded from the scope of the Design Analysis.

1.2.2.5 Hardware Modifications

Modifications to the hardware of the Honeywell Model 6180 system and its peripheral devices were specifically excluded from the scope of the Design Analysis. No hardware modifications are in fact required.

1.2.3 End Product of the Design Analysis

This document is the end product of the Design Analysis. It describes the requirements for operating a Multics system in a restricted multi-level security mode for Secret and Top Secret users working in a closed secure environment.

The requirements are translated into a functional design of modifications to the Multics system needed to provide this restricted multi-level security operation.

In addition, the user limitations and potential operational/administrative problems internal and external to the system are outlined.

This document is expected to be the basis of the proposal for the implementation phase of the security controls as described in CORL Item A010 of Air Force/Honeywell contract number F19628-73-D-0087.

2. APPLICABLE DOCUMENTS

2.1 Air Force/Honeywell contract number F19628-73-D-0087

This contract provides the authority for the Security Design Analysis. The documentation requirements for the final report and the allowed deviations from the format are specified in this contract. The AFDSC Multics RFP No: F19628-73-R-0024 is included in the contract. Annex 5-1 of that RFP defines the primary requirements for Multics security controls.

2.2 DoD 5200.1-R Information Security Program Regulation

Describes the military security system and the responsibilities of personnel who fall within its jurisdiction.

2.3 AFR 205-1 Information Security Program (USAF)

Implements DoD 5200.1-R

2.4 DoD 5200.28 Department of Defense Directive, Security Requirements for Automatic Data Processing (ADP) Systems

Defines the security requirements for processing classified data on an ADP system (See 2.5).

2.5 DoD 5200.28-M Manual of Techniques and Procedures for Implementing, Deactivating, Testing, and Evaluating - Secure Resource Sharing ADP Systems.

This is the manual which outlines the details of the general requirements specified in DoD 5200.28.

DoD 5200.28 and DoD 5200.28-M were not identified as mandatory documents to be followed for the Multics system at the time the AFDSC RFP was issued. However, the requirements have been met as closely as possible in designing the Multics Security Controls described in Section 3.

2.6 MIL-STD-483 Appendix VI Para 60, "Computer Program Configuration Item Specification"

Air Force suggested documentation format specification for the final report of the Design Analysis.

This standard has been followed for content and general order of presentation. Deviations from the strict format of the CPCI specification were authorized by the contract (Paragraph 2.1). Section 3 of the standard has been expanded in this document to provide a form of presentation better suited to the material.

2.7 Honeywell Multics documentation

The following documents are mentioned here as a source of background information concerning the Multics system.

Multics Programmers' Manual
- Introduction (AG90)
- Reference Guide (AG91)
- Commands and Active Functions (AG92)
- Subroutines (AG93)
- Subsystem Writer's Guide (AK92)

Project Administrator's Manual (AK51)
System Administrator's Manual (AK50)
PL/I Language Manual (AG94)
Multics Virtual Memory (AG95)
The Multics System (AK27)

The order numbers given above (e.g. AG90) should be specified when ordering these cocuments from Honeywell.

2.8 General references

The following documents are mentioned here as a source of background information concerning computer security and, in particular, military computer security.

Multics Evaluation, J. P. Anderson, ESD-TR-73-276, Electronic Systems Division (AFSC), L. G. Hanscom Field, Bedford, MA, October 1973.

Design and Certification Approach: Secure Communications Processor, P. S. Tasker and D. E. Bell, MTR-2436, The MITRE Corporation, Bedford, MA.

Secure Computer Systems: Mathematical Foundations, D. E. Bell and L. J. LaPadula, ESD-TR-73-278, Vol I, Electronic Systems Division (AFSC), L. G. Hanscom Field, Bedford, MA, November 1973.

Computer Secure Research and Development Requirements, S. B. Lipner, MTP-142, The MITRE Corporation, Bedford, MA, February 1973.

Preliminary Notes on the Design of Secure Military Computer Systems, R. R. Schell, P. J. Downey, and G. J. Popek, MCI-73-1, Electronic Systems Division (AFSC), L. G. Hanscom Field, Bedford, MA, January 1973.

Concept of Operation for Handling I/O in a Secure Computer at the Air Force Data Services Center (AFDSC), E. L. Burke, ESD-TR-74-113, L. G. Hanscom Field, Bedford, MA, October 1973.

3. SECURITY REQUIREMENTS FOR AIR FORCE DATA SERVICES CENTER

The Air Force Data Services Center has a requirement to provide ADP resources and services for the processing of unclassified through Top Secret data to support Headquarters USAF and the Office of the Secretary of the Department of Defense. In providing this capability, the AFDSC is responsible for the security of the classified data processed on their computer systems.

Most contemporary shared computer systems are not secure because security was not a mandatory requirement of the initial hardware and software design. The military has achieved reasonably effective physical, communication, and personnel security. Hence, the primary computer security problem is that of information access controls in the operating system and supporting hardware. Essentially, an effective means for enforcing very simple protection relationships (e.g. user clearance level must be greater than or equal to the classification level of accessed information) is needed; however, solutions to some of the more complex protection problems such as mutually suspicious processes are not required.

In current practice at AFDSC, computer security is achieved by dedicating an entire computer system to users cleared to a particular security level. This approach results in poor utilization of computer resources, and hence, high costs for data processing.

Providing a two-level security operating mode on the Honeywell 6180 Multics System will be the first step toward fully utilizing the resources of a single computer system serving a user community with multiple-level security requirements.

The decision to design and implement a two-level security system for the Air Force Data Services Center is predicated on our capability to provide those security controls that will reduce the risk of release of Top Secret information to Secret users to an acceptable level. No claim is being made as to the ability of the security system to withstand penetration attempts. The approval test that the system will be subjected to prior to its installation will only demonstrate the existance of security controls. It is

anticipated that the efforts to augment the security of the Multics System combined with the limitation imposed on the operation of the system within the AFOSC controlled environment will provide an operationally acceptable assessment of risk.

3.1 System Operating Environment Definition

3.1.1 Hardware/Software Interface

The central processing unit used is the Honeywell Model 6180. The operating system is Multics, with such modifications and extensions as result from this Security Design Analysis and the system programming task that will follow.

A full description of the Honeywell 6180 hardware and the Multics software is beyond the scope of this document. The interested reader is referred to the publications listed in Section 2.7 for such detailed descriptions.

3.1.2 User Interface

The user interface is the appearance the system presents to the user. To the greatest degree possible, this appearance will remain the same as current Multics.

Functions available to the user will be identical to current Multics where feasible, and equivalent in most other cases.

3.1.3 Definition of AFDSC Controlled Environment

The central computer facility will be a Top Secret controlled area.

All remote terminal areas will be physically protected to the Top Secret level even though they may be used as Secret controlled areas.

The communications between the central computer facility and all remote terminal areas will be via Top Secret encrypted data lines.

Top secret clearances will be required for all persons (operators, system programmers, system maintenance personnel, field engineers and others) who need physical access to the central computer facility; or any hardware,

data lines or terminal connections in the remote terminal areas; or data and control lines between the central computer facility and the remote terminal areas.

All programmers, analysts, users or persons who are registered to use the Multics system at AFOSC will have either a Secret or a Top Secret clearance.

All I/O operations will be performed by central site operating personnel. No user will be permitted to mount his own tapes, disks or other media.

3.1.4. Definitions

access

The ability and the means to approach, communicate with (input to or receive output from), or otherwise make use of any material or component in an ADP System.

In the military security system, a person may be granted access to an object only if his clearance level is greater than or equal to the classification level of the object; his clearance category set contains all categories in the category set of the object; and he has the proper "need to know" in reference to the object.

ADP (Automatic Data Processing)

An assembly of computer equipment, facilities, personnel, software and procedures configured for the purpose of classifying, sorting, calculating, computing, summarizing, storing, and retrieving data and information with a minimum of human intervention.

anonymous user

An anonymous user is an unregistered user of the Multics system whose personid (see below) is "anonymous"; in other words, his personid is unknown to the system. An anonymous user may or may not be required to furnish a password in order to gain access to the system.

branch

A branch is a component of a directory which describes an immediately inferior segment or directory.

Interprocess communication (ipc)

Interprocess communication is a facility which allows one process to communicate with another in a controlled manner. Both the sending and receiving processes must adhere to a specified protocol.

"level"

This term is used frequently as an abbreviation for the level/category combination which describes a clearance or a classification. Thus the "level" of a process is the clearance of the process and the "level" of a segment is the classification of the segment.

Multi-Level Security Mode (see also Two-Level Security Mode)

A mode of operating under an operating system (supervisor or executive program) which provices a capability permitting various levels and categories or compartments of material to be concurrently stored and processed in an ADP System. In a remotely accessed resource-sharing system, the material can be selectively accessed and manipulated from terminals by personnel having different security clearances and access approvals. This mode of operation can accommodate the concurrent processing and storage of (a) two or more levels of classified data, or (b) one or more levels of classified data with unclassified data depending upon the constraints placed on the systems by the Designated Approving Authority.

Operating System (O/S)

An integrated collection of service routines for supervising the sequencing and processing of programs by a computer. Operating systems control the allocation of resources to users and their programs and play a central role in assuring the secure operation of a computer system. Operating systems may perform debugging, input-output, accounting, resource allocation, compilation, storage assignment tasks, and other system related functions (Synonymous with Monitor, Executive, Control Program, and Supervisor).

personid

The registered name of someone who is authorized to use the system. It is usually constructed from the last name (surname) of the person.

process

A process is the active agent of the user on Multics and (in the security system) has a clearance which may not exceed the user's clearance. The lifetime of a process normally corresponds to a user's terminal session and is described internally by an address space and a point of execution. Both the address space and the execution point are dynamic over the life of the process.

projectid

The registered name of a project which has an account on the system.

Remotely Accessed Resource-Sharing Computer System

A computer system which includes one or more central processing units, peripheral devices, remote terminals, and communications equipment or interconnection links, which allocates its resources to one or more users, and which can be entered from terminals located outside the central computer facility.

segment

A segment is a logical unit of storage on Multics. It roughly corresponds to a file stored on a disk pack and accessible to a user. The segment is the smallest element of supervisor access control in the Multics storage system.

Two-Level Security Mode

A mode of operating a computer system which provides a capability permitting Top Secret and Secret data to be concurrently stored and processed in an ADP System. This mode is more restricted than the multi-level security mode in that only Top Secret and Secret cleared users will be permitted to access the system. No unsecure terminals will be connected to the system. Software, hardware, administrative, and physical controls will provide the safeguards to assure the integrity of the classified data processed.

user

An instance of a person logged into the system on a project. A user is identified by a userid.

userid

A table entry which would describe a user (e.g. an access control list entry). A userid consists of "personid.projectid.tag," where tag is normally "a" for an interactive user, "m" for an absentee user, and "z" for certain system daemons. The userid is also called the "principal identifier" or "group_id" of the user.

3.2 APPLICATION OF SECURITY CONTROLS TO MULTICS

Each person registered on Multics is known to the system by his name (personId) and has a password to authenticate his identity. The authentication data for a personId must include the person's system-recognized clearance.

Each user of Multics, as identified by his userId (person-project combination), is associated with a Multics process. Each Multics process must have a clearance which is equal to or less than the clearance of the person associated with the process and must remain constant for the life of the process.

Access control is generally described as a subject attempting to access an object through an intervening reference monitor. The reference monitor checks, each and every time a subject attempts to access an object, to see if the subject has the proper authorization to perform the desired operation (e.g. read, write, execute, append, modify, delete). In Multics, a process is the only subject which can make a reference to any object. The set of objects are segments, directories, branches, I/O channels and interprocess communication messages. Each object must have a classification level and category set associated with it.

In Multics, the reference monitor which validates each reference to an object is the "ring 0" supervisor in conjunction with processor hardware protection mechanisms. Within the protection ring scheme supported by the Honeywell 6180 processor, ring 0 is the most privileged and most protected ring of operation. All access control decisions are made within ring 0. Each time a process attempts to gain access to an object, the clearance of the process is compared with the classification of the object and access is either granted or denied in accordance with rules designed to emulate the military security system. In addition to classification, certain objects such as segments and directories have an associated access control list which specifies persons having need to know authorization as in the military security system.

When the classification of two objects is compared, four relationships are possible:

less than

equal

greater than

isolated

The classification of object 1 is considered "less than" the classification of object 2 if:

1. The level of object 1 is numerically less than or equal to the level of object 2; and

2. The category set of object 1 is a subset of the category set of object 2; and

3. The classification of object 1 is not equal to the classification of object 2.

The classifications of two objects are considered "equal" if:

1. The levels are numerically equal; and

2. The category sets are identical.

The classification of object 1 is considered "greater than" the classification of object 2 if:

1. The level of object 1 is numerically greater than or equal to the level of object 2; and

2. The category set of object 2 is a subset of the category set of object 1; and

3. The classification of object 1 is not equal to the classification of object 2.

The classifications of two objects are considered "isolated" if the category sets are isolated.

The "minimum" of several classifications is defined as:

1. The numerical minimum of the levels; and

2. The intersection of the category sets.

In order for a person to access information, the military security system requires that the clearance of the person be

greater than or equal to the classification of the information. A sufficient condition for satisfying this requirement within the computer system environment is the enforcement of the following two rules:

1. A process having clearance n may not "read up," i.e. read an object having a classification greater than n.

2. A process having clearance n may not "write down," i.e. write an object having a classification less than n.

With these two rules enforced, it is impossible for any process to extract information from an object of higher classification or to transfer information from an object of higher classification to an object of lower classification. Hence, no compromise of classified information can occur. This principle is known as the "fixed level property." A further restriction is also desirable which forbids a process to write in an object of higher classification whenever writing can be used to destroy information. In order to provide some protection against sabotage, "write up" operations must not be permitted for such objects as segments, directories, and branches.

It is important to recognize that the rules described above represent a sufficient, but not a necessary condition for achieving security. Although the fixed level property restrictions will be strictly enforced for all user processes, they will, in certain circumstances, be applied interpretively for trusted system processes. In no circumstances, however, will security be violated, because trusted system processes must operate correctly.

The individual user must be able to specify which users should have "need to know" for a given segment or directory by use of the Access Control List. The mode of access (e.g. read, write) allowed to a process by the current Multics Access Control List must be further restricted to ensure compliance with the fixed level property rules. In other words, the fixed level property rules must take precedence over the Access Control List.

Information transmitted between hardware modules must be carefully controlled by the system and no user should be able to directly affect the action of an active module (except for the CPU). Furthermore, no user process should be able to execute any programs which would perform external I/O to any device other than his terminal.

The system can be logically divided into two environments: internal and external. The internal environment is totally controlled by the system. This includes: processors, memory, disk drives, I/O multiplexers, bulk store,

communication processors, and tape drives used for system functions.

The external environment can be directly influenced by the actions of a process. This environment includes: terminals, line printers, card readers, card punches, non-system tape drives, and other devices in the I/O class not used for system functions.

To provide a "secure" pipeline between the internal and external environments, a trusted process must perform the actual information transfer on behalf of the user. This will further ensure that failures or "software bugs" will not be exploited by a user. The terminal must be the only exception to this rule and this exception is only made for the sake of efficiency.

Whenever possible, new or modified operator interfaces supplied with the security control features will be designed to provide extra aids or simplicity in structure to help the operator avoid mistakes which could become security violations.

Security and administrative functions should be separated to ensure that the System Administrator will not make security-related decisions and to avoid burdening the Security Officer with purely administrative decisions.

The security controls must be designed so that: the system is easy to use; the users are encouraged to properly classify data (rather than over-classify); the least possible amount of current Multics functionality is sacrificed; and the current user interface is maintained wherever possible.

All high-level security-related actions performed within the system should be audited to ensure user responsibility and to provide early warning of any subversion attempts, misuse of the security controls, or actions which could lead to compromise.

All revisions to the system must be carefully checked to minimize the possibility of "bug fixes" or new "features" causing the system to behave incorrectly, especially insofar as security is concerned.

3.3 PROCESS CLEARANCE ASSIGNMENT

3.3.1 Requirements

A Multics process is uniquely associated with a person who is registered to use the system and a project to which that person may charge his system expenses.

When a process is created for a user, a clearance will be established for the process. This clearance must not be changeable by request for the life of the process. It is the process clearance which will be used to determine a user's authorization to access classified information in the system.

To provide a degree of flexibility and administrative control, the clearances of several entities must be stored on the system.

The data associated with a personid (the system unique identification for the person) must contain the clearance of the person. Similar clearance data must be associated with each projectid. In addition, the data which describes the limitations of a person on a given project must have clearance data.

The clearance to be assigned to a process must be determined as follows:

1. No process will be created for a given userid, i.e. a given person on a given project, with a higher clearance than the minimum of the person's clearance, the project's clearance, and the person's clearance within the project.

2. No user should be able to create a process with a higher clearance than the maximum clearance of his terminal.

3. A user must be able to request a process with a lower clearance than the minimum of his userid and terminal clearance.

4. A user must be able to specify a default login clearance (no higher than his personid clearance).

Only the System Security Officer (SSO) must be able to assign clearances for a personid or a projectid. If the SSO lowers the clearance of a personid, the user's process must be forceably terminated if he has an active process with a clearance greater than the downgraded clearance of the personid.

Each user must be told his process clearance at the beginning and normal termination of the process. In this way, the user is made explicitly aware of his level of operation. Hence, mistakes such as placing Top Secret information in a Secret file are unlikely to occur, and if they do occur, are likely to be detected before any harm can result.

By use of a command, each user should be able to request that the clearance of his current process be typed on his terminal.

The names associated with a "level" should be set by the installation.

3.3.2 Design Approach

The system control process uses three tables to verify that a user should be logged in.

1. The Person Name Table (PNT) contains an entry for each personid on the system.

2. The System Administration Table (SAT) contains an entry for each projectid on the system.

3. The Project Definition Table for the users project (Proj.pdt) contains an entry for each personid allowed to used the project. There is one project definition table for each projectid.

Each of these tables will be modified to hold clearance level and category set data for each entry. The system control process will check this clearance data to determine the maximum clearance for a userid.

A new table, called the Peripheral Control Table (PCT), will be used by the system control process to check the maximum clearance of the terminal being used by a person attempting to log in. Since terminals will be "hard wired" to the system at AFDSC, each terminal can be uniquely identified by an associated channel number. In the general case, there may be crypto-dial-up terminals. However, in that case, the crypto units will provide the unique terminal identification. As an extra check, the answerback code

received from a terminal will be compared against its "registered" answerback code. This answerback test will be useful in detecting mistakes, as well as malicious tampering, involving communications lines and terminals.

At login time, the user will be given a process with a clearance no higher than the minimum of the clearances from the PNT, SAT, Proj.pdt, and the PCT. The default login clearance for each user will initially be the lowest possible clearance, i.e. unclassified. A new login option will be supplied to permit a user to change this default. Also, another new login option will be provided which allows the user to specify a particular clearance for a given login.

An attempted login may be rejected for the following reasons:

1. illegal login word

2. incorrect personid or projectid

3. incorrect password

4. incorrect level option

5. unrecognized login option

These rejected login attempts will be recorded for audit purposes. In addition, if a user attempts to use a terminal with a maximum clearance greater than the personid clearance from the PNT, a message will be sent to the operator, since this will indicate a breach of physical security. The clearance of the process will be stored in the process initialization table (pit) and in the ring 0 process data segment (pds) of the process to ensure that it is unforgeable for the life of the process.

The Project Administrator will be able to specify for a user on his project a lower maximum clearance than authorized in the PNT and SAT, if this ability is granted by the SSO.

Person and project responsibilities of the System Administrator will remain the same as on the current Multics systems. When a new user or project is added to the system, the maximum clearance will be set to unclassified. Only the SSO will have access to the commands to update clearances in the PNT, SAT and PCT.

Anonymous users should not normally be permitted on the system since password authentication is not always required for them. Where passwords are required for anonymous users, these passwords are controlled by project administrators

rather than the SA or the SSO. If, at any time, anonymous users are permitted on the system, they will always be given unclassified processes.

Absentee processes will be created at the level of the requesting process unless an option is specified. A user will not be able to create an absentee process with a clearance which is lower than his current process clearance, since the passing of arguments to the absentee process would constitute a write-down operation.

A new_proc option will be added to allow a user to upgrade/downgrade his level of operation. When no option is specified, the default level for the new process will be the current level. (The same will be true for abnormal termination of a process).

The system process_overseer_ procedure will identify the "level" of a process created for a user by printing the "level" name on his terminal. (This cannot be defeated.) The same message will be printed by terminate_process_ for normal process termination.

Installation parameters will be used to store the character strings used to identify each classification level and category. The system assumes that the names used for levels and categories are unclassified.

Each user will be able to execute a command which will print the "level" of his process on his terminal based on the data in the "pit."

3.3.3 Potential Security Problems

The following areas will become security problems only if the non-malicious user assumption of Section 3.1.4 is violated.

The ability for a user to enter an absentee request of an equal or higher "level" than his process clearance is one way for a Trojan Horse to gain control of a user's access permissions without the user noticing excessive processor usage or real time delays within his current process. If this happens, a need to know violation or sabotage can occur very easily, but the only means for compromise would be through the quota path on directories which has a very low transmission rate. (See Section 3.7.4)

By providing a means for a user to change his "level" of operation through program control (new_proc with level option), a Trojan Horse could set itself up as the program to be called when a user attempts to change to a new "level"

process. An elaborate Trojan Horse could totally simulate system action for new_proc to fool the user into thinking he is operating at a higher level. Now if the user attempts to input classified data, the Trojan Horse could, by simulating the entire user interface, cause the user to put the classified data into a segment with a lower classificatior. This problem can be solved by only allowing a user to "new_proc" to the same or lower "level."

In a similar manner, a user may write his own "logout -hold" command to fool the next user of the terminal into thinking he is talking to the system instead of the previous user's process. This could allow a malicious user to capture the password of another user, thus permitting sabotage and need to know violations. (See Section 3.4.1.) Also, the user environment simulation described above could be used here. The solution to this problem is to require the terminal to be powered off by each user before attempting to login. (This can be handled several ways. The choice is up to the site manager.)

Solutions exist to all of the above potential problems. However, given the low expectation of occurrence of these problems, the required sacrifices in user convenience were felt to be unwarranted within the assumed benign environment.

3.4 PASSWORD CONTROL

3.4.1 Description

The Multics access control mechanism depends on several important factors. First and foremost is the notion of an unforgeable "user name" which identifies the access rights of a Multics process; the entity which performs all tasks on behalf of the human user. A Multics "user name" is called a principal identifier, and consists of three components: Person, Project, and Tag. The Person component uniquely identifies a registered user of Multics. The Project component identifies a registered project, and Tag is presently derived from the type of process (i.e. interactive, absentee, or consoleless daemon).

In order for Multics to successfully enforce access controls, it must be possible to uniquely and positively identify each user at login. This is presently accomplished by assigning each registered person his own password, and at each login, requesting his password for verification purposes. If the password stored by Multics matches the password given by the user, Multics assumes the user is valid, and creates a process with the principal identifier (userid) of the user. If, after giving the user several chances (to allow for typing mistakes), a correct password has not been received, Multics refuses the login.

Clearly, the password is a vital part of the access control mechanism, and as such, must be carefully protected by both the user and the system. If a person could guess (by whatever means) another user's password, that person would himself be able to log in as the other user. It should be noted, however, that due to physical security controls at AFDSC, the compromise of a password cannot result in the compromise of classified information. A person who learns another person's password will not be able to log in with the same clearance as the owner of the password unless he, himself, has an equal or higher clearance which affords him access to a terminal of equal or higher classificatior. Therefore, password compromise can, at worst, result in sabotage or need to know violations.

3.4.2 Requirements

The present "work factor" needed to guess a person's password is not high enough, due to the ability of a user to choose his own password. Therefore, it is a requirement that the system assign passwords. (The passwords could have been distributed manually, but that was felt to be too burdensome for the system administrator.)

To provide the ability to control the "age" of a password (how long it has been in use by a user), it is a requirement that the system be able to force a user to change his password at pre-determined intervals.

To be able to recover from a password breach, it is a requirement that the System Security Officer be able to force some or all users to change their passwords.

3.4.3 Design Considerations

Since all users must go through the login ritual, every attempt will be made to "human engineer" this area of the system. The passwords generated by the system will be designed to be pronounceable and therefore, easy to remember.

3.4.4 Chosen Approach

After the identity of the user has been authenticated by the login procedure, the system will warn the user if it is time to change his password. To force the user to change his password within an installation-parameter grace time, the user will be locked out if he exceeds the grace time. To properly handle persons who login infrequently, the grace "time" will actually be implemented as a grace number of logins.

The system generated passwords will be based on English digraph frequencies since such words are more pronounceable, and thus more easily remembered, than random strings of characters.

Since passwords must be treated as classified information, the system will prefix the printing of a new password with the label "confidential."

To ensure that the user understands the new password and that it was printed correctly, the user will be required to echo it at login time.

The SSO will be able to set the interval at which users must change their passwords.

The SSO will be able to force a user to change his password.

Incorrect login attempts will be audited (see Section 3.16).

3.4.5 Examples

```
1*      login Whitmore -charge_password
2       Password:
3*
4       Confidential: New password is "abcodo."
5       New Password:
6*
7       Password changed.
```

Lines marked with "*" are typed in by the user. The terminal does not print passwords typed by the user.

In the first example, Whitmore requests that his password be changed. The system requests his current password and assigns him a new one. The user is requested to enter his new password for verification. If both passwords were typed correctly, he will be logged in and his password will be changed within the system. If either password was incorrect, the entire login would be incorrect and the user would have to try again.

```
1*      login Whitmore
2       Password:
3*
4       You must change your password within 3 logins
```

In the second example, Whitmore is notified that his password must be changed within the next three logins. If he fails to change his password, he will be locked out. The user may login, even if he has been locked out, by changing his password.

3.5 INFORMATION CHANNELS BETWEEN PROCESSES

The fixed level property rules defined in Section 3.2 are designed to restrict the passing of information between processes. These restrictions must be applied to all information channels, i.e. to all mechanisms within the system which enable processes to exchange information. Certain mechanisms such as shared segments and the interprocess communication facility are deliberately provided to serve as information channels. Other mechanisms such as segment names and access control lists are intended to serve different purposes, but could be misused as information channels by processes attempting to compromise information. Hence, all information channels must be identified and, where necessary, additional access checking must be provided in order to enforce the fixed level property rules.

3.5.1 Segment Sharing

A shared segment is the most natural channel for two processes to exchange information. For a process with a clearance P, the system will systematically remove the "write" permission on any segment whose classification is lower than P, and all permissions on any segment whose classification is higher than P. It is therefore impossible for a process to "write down" or to "read up."

More detail can be found in section 3.6 - "Access to Segments."

3.5.2 Directories

Directories are another channel through which processes can exchange information. Each data item contained in a directory is assigned a specific classification (as described in Section 3.7). Ring 0 primitives in charge of manipulating directories will provide additional checking by which they will systematically refuse to perform a request if it would result in a "write down" or a "read up."

Unfortunately, however, a number of directory items such as quota used, date-time modified, and date-time used are changed not by explicit request, but rather as a side-effect of some action performed outside the directory. For example, the quota usec count stored in a directory can be increased by growing the size of an inferior segment. Information channels of this type present rather unusual problems. Solutions to these problems as well as other details of directory access control are discussed in "3.7 Access to Directories."

3.5.3 Interprocess Communication

Using the Interprocess Communication (IPC) facility, a process can send a 72-bit message to another process. The IPC facility will provide additional checking by which it will systematically refuse to send a message that would result in a "send down." "Send up" will be permitted for IPC since this is not a means of sabotage. The enforcement of the security will be done in ring 0.

3.5.4 Message Segments

In the current Multics System, message segments are ring one segments, manipulated by a ring one module called the Message Segment Facility (msf). The implementation of the msf is such that a process needs the "read" and the "write" capabilities in ring one on a message segment in order to be able to put a message in it or to extract a message from it. It follows that, if the msf is used within the security controls, communication between processes through message segments will be restricted to processes of identical clearance. This restriction has been accepted.

As far as security is concerned, message segments will be treated the same as any other segment by the ring 0 supervisor and one can repeat what was said for segments in general: no read up or write down on a message segment will be permitted in a user process. However some system processes, in some special cases and in a controlled manner, will have to bypass the fixed level property restrictions on message segments. However, in no circumstances will security be violated.

In the current Multics system, all user processes that request a service from a system process send their request through a message segment. It follows that, wherever the current system uses one message segment to queue user requests for a system process, it will be necessary to provide one message segment for each existing classification.

An alternative approach would have allowed security rules to be enforced in ring one rather than ring 0. In this scheme, ring 0 would grant read and write access for message segments to processes of all clearances. Each individual message stored in a message segment would be "classified" by the msf at the clearance level of the sending process. The msf would only permit extraction of a message by a process having a clearance higher or equal to the classification of the message. However, this would bring the msf plus all other ring one procedures within the security perimeter, thereby making the task of certification more difficult.

3.5.5 Summary

It is important to understand that of the several information channels described above, shared segments are the only channel through which classified information would routinely be stored and passed. IPC messages and directory items such as segment names or access control lists would not normally be used to transmit or store classified data. (All segment names are assumed to be unclassified so that they may appear in unclassified accountability forms for printed output. See Section 3.10.) Hence, from a practical standpoint, the assigning of correct classifications to segments by users and the addition of fixed level property access checking for segments is sufficient to prevent a single malicious user from directly compromising classified information.

The other information channels do not become a serious problem until one considers the possibility of two (or more) processes cooperating in an effort to compromise information. This cooperation could take one of two forms. First, two malicious users might directly conspire to compromise information. Second, a nonmalicious user might unknowingly employ a Trojan Horse program supplied by a malicious user. (See Section 1.2.2.2.) The case of two users conspiring to compromise information is actually more of a "people" problem than a computer system problem. Even if no effort were made to secure those information channels not normally used to store or transmit classified data, conspiring users would probably still find it easier to pass information outside the system. Therefore, the Trojan Horse attack is really the only form of attack for which information channels other than segments are essential.

The design presented in this report is directed to eliminating all read-up and write-down information channels. The elimination of all known read-up channels prohibits a malicious user from directly accessing classified information which he is not legitimately cleared to see. Hence, a malicious user must resort to "setting a trap,"

i.e. he must create a Trojan Horse program with the hope that an unsuspecting user having a higher clearance will call it. Although a general solution to the Trojan Horse problem is beyond the scope of this design, the elimination of write-down channels can considerably reduce the threat represented by the Trojan Horse form of attack. A write-down channel is the only means by which a Trojan Horse program can actually compromise information. Therefore, the elimination of all write-down channels can effectively prevent compromise, although sabotage and need to know violations would still be possible. With one exception, all explicit write-down channels within the Multics system have been eliminated in this design. The quota used channel is the single exception. Not only does this channel have a very low transfer rate, but also, any significant use of this channel can be easily detected through auditing. (See Section 3.7.3 for details.)

3.6 ACCESS TO SEGMENTS

3.6.1 Requirements

Every segment must have a classification defined by a level and category set. This classification applies to all data contained within the segment.

For each segment there must exist a list of persons having need to know access to the contents of the segment.

The sharing of segments among processes must be controlled so as not to violate the fixed level property rules.

3.6.2 Design

The classification, i.e. level and category set, of a segment will be permanently recorded in its branch. For reasons explained in Section 3.7, the classification of a segment must equal the classification of its parent directory. This implies that the classification of a segment will always equal the clearance of the process which created it, since a process can only append a branch to a directory if its clearance equals the directory classification.

As is already the case in Multics, an access control list (ACL) will be associated with every branch. Each ACL entry contains a userid and access mode. The access mode describes the types of access (e.g. read, execute, write) permitted the associated user. Hence, the ACL will be used to control need to know access to a segment.

In accordance with the fixed level property, write down and read up operations on segments will be prohibited. Also, in order to prevent sabotage, write up operations on segments will be prohibited. With these restrictions enforced, sharing of segments among processes having different clearances cannot compromise information.

The access permitted a given process to a given segment will be computed as follows. If the clearance of the process is lower than the classification of the segment, the process will be given null access to the segment. If the clearance of the process equals the classification of the segment, the

process will be given whatever mode of access, if any, is specified in the ACL. If the clearance of the process is higher than the classification of the segment, the process will be given the mode of access specified in the ACL minus write permission.

In order to reference a segment, a process must first "initiate" the segment into its address space. At initiation time, the access computation described above will be performed to determine if the process has any access to the segment. If so, the segment will be added to the address space of the process. Thereafter, all references to the segment will be validated by the processor hardware. Each segment fault taken by the process on the segment will force access to be recomputed by the above method.

3.6.3 Implications

The rules governing access to segments, while satisfying security requirements, have certain curious implications worth noting. A problem arises over the fact that for each user there typically exists a number of corresponding writeable data segments (e.g. mailboxes, console message segments, abbrev profiles, pmotd files). Conceptually, it makes little sense to segregate the functions of these segments according to process clearance. Nevertheless, these segments must be assigned a specific classification and hence, will be writeable by a process at one clearance level only. As a result, the user who operates at more than one clearance level must sacrifice a certain amount of flexibility and convenience in sending and receiving mail, creating abbreviations, updating pmotd files, etc.

3.7 ACCESS TO DIRECTORIES

3.7.1 Classification of Directory Information

Every directory has a classification defined by a level and a category set.

The classification of a directory cannot be less than the classification of its parent directory. This restriction is necessary in order to eliminate a write-down information channel using directory names. Suppose, for example, that an unclassified directory were permitted to exist in the hierarchy below a Secret directory. A Secret process could change the name of the Secret directory, thereby also changing the pathname of the unclassified directory. This action could, of course, be detected by an unclassified process. Therefore, it is necessary that a directory and, for that matter, a segment, have an equal or greater classification than its parent directory. This rule is hereafter referred to as the "non-decreasing classification rule." For reasons explained below, the classification of a segment is further restricted to be equal to that of its parent directory.

As with segments, a directory will initially receive the same classification as its parent directory. However, a special "upgrade" operation will be available which permits a user to raise the classification of a directory. It is required that a directory be empty in order to be upgraded. Otherwise, after upgrading, inferior segments or directories would stand in violation of the non-decreasing classification rule. If the entire subtree of a directory were upgraded, a potential for unwanted overclassification would exist. (Also, implementation would be difficult.)

Several problems arise with respect to the branch of an upgraded directory. Since, as described so far, such a branch is contained in a superior directory of lower classification, a user having access to an upgraded directory would not be permitted to modify its branch. This restriction would be very inconvenient in practice. However, a more serious problem is posed by the fact that a user having access to an upgraded directory would be able to implicitly modify its branch. For example, by increasing the size of an upgraded directory, one could change the

current length attribute in its branch. This constitutes a write-down information channel.

In order to eliminate the above problems, an individual branch will have the same classification as the segment or directory which it describes, rather than that of its containing directory. Only upgraded branches are actually affected by this new definition, since non-upgraded branches will still have the same classification as their containing directory anyway.

The classification of a branch applies to all data items within the branch except for the branch names. These names retain the classification of the containing directory. Names are separated from the branch in this way in order to avoid creating still another write-down information channel. If the name of an upgraded directory could be modified by a process at the "level" of the branch, then a lower-level process could detect such modifications by adding names to non-upgraded branches in the same directory and observing whether name duplications occurred. Hence, branch names can only be modified by a process at the "level" of the containing directory.

3.7.2 Explicit Operations on Directories

Whenever the supervisor is explicitly requested to perform an operation on a directory, a check will be made to ensure that the user has the right to perform the operation according to the current Multics access control rules and the new fixed level property rules. In particular, the supervisor will refuse any request that would result in a "read up" or a "write down"; it will also refuse all requests that could result in sabotage by "writing up."

Operations that would return to the caller any part of a directory having the same classification as the directory itself will be executed only if the clearance of the process is equal to or higher than the classification of the directory. Examples of these operations include listing a directory, listing the initial ACL, and reading the quota.

Operations that would modify any part of a directory having the same classification as the directory itself will be executed only if the clearance of the process is equal to the classification of the directory. Examples of these operations include adding or deleting entry names, changing the initial ACL, and creating a new branch (since a branch is originally created with the classification of its containing directory). The deletion of branches, both upgraded and non-upgraded, is also included in this category since it involves the deletion of branch names. Note,

however, that this does not constitute a means of sabotaging an upgraded directory since it is required that an upgraded directory be empty in order to delete it. Subtrees are always deleted in a bottom-up fashion. Therefore, a user able to delete an empty upgraded directory will not be able to delete that same directory when there exist inferior segments or directories to which he has no access.

Operations that would return to the caller any part of a branch, other than the entry names, will be executed only if the clearance of the process is equal to or higher than the classification of the branch itself. Examples of such operations include reading the branch status, reading the ACL, reading the ring brackets, reading the bit count, reading the date-time used or modified.

Finally, operations that would modify any part of a branch other than the entry names will be executed only if the clearance of the process is equal to the classification of the branch. Examples of such operations include changing the ACL, changing the bit count, changing the maximum length, and changing the safety switch.

The "movequota" operation is unique in the sense that it modifies two directories at once, one immediately inferior to the other. A problem arises when quota is moved to or from an upgraded directory. To do this, a process is required to modify two directories of different classifications which is normally not permitted. Since writing down must be prohibited, a process at the "level" of an upgraded directory cannot be allowed to move quota between that upgraded directory and its parent directory. Therefore, the movement of quota to or from an upgraded directory will be performed only by a process at the "level" of the parent directory. (Modify permission on the ACL of both directories will still be required.) The fact that a lower-level process will be able to withdraw quota from an upgraded directory constitutes a mild form of sabotage which can only temporarily impede a higher-level process, but cannot destroy or compromise information. This is not felt to be a serious problem since this could be auditable and quota can easily be restored. The alternative of not allowing quota to be withdrawn from an upgraded directory except by special action of the SSO is considerably less attractive.

The new upgrade operation for directories is also rather unique. Since it involves modifying an element of a branch, it can only be performed by a process at the same "level" as the branch. In addition, the directory to be upgraded must be empty as mentioned above. Furthermore, for reasons to be explained shortly, the directory to be upgraded must have a terminal quota.

3.7.3 Implicit Operations on Directories

As described above, additional access checking can be performed for all explicitly requested directory operations so as to comply with the fixed level property rules. Unfortunately, however, there exists a class of implicit directory operations which present a more difficult problem. An implicit directory operation is basically a side-effect of some action performed outside the directory. One such operation, the changing of the current length attribute, has already been discussed. Three other implicit directory operations, which are the changing of quota used, date-time used (dtu), and date-time modified (dtm), still cause problems within the directory access scheme described thus far. These problems are discussed below.

3.7.3.1 The Quota Used Problem

Changing the number of pages used by a segment or directory causes the "quota used" number to be incremented or decremented in all superior directories up to and including the nearest superior directory having a terminal quota. If this chain of superior directories includes one or more directories of a lower classification than the segment or directory being modified, then a write-down information channel exists. There are three methods of performing write-down operations on this information channel: 1) changing the number of pages used by segments in an upgraded directory; 2) increasing the pages used by an upgraded directory itself; and 3) increasing the pages used by the parent of an upgraded directory due to an increase of the upgraded branch.

The First Method

Changing the length of segments to reflect the "quota used" up the chain of superior directories is the most flexible method of using this information channel. However, this facet of the problem can be blocked by requiring that a segment have the same classification as its parent directory and that every upgraded directory have a terminal quota. In this way, the pages of a segment are always charged to the quota of a superior directory having the same classification as the segment. Hence, one cannot pass information down merely by changing the size of a segment and causing the "quota used" number to change in some superior directory.

The Second Method

Pages of an upgraded directory itself are charged to a superior directory of lower classification. This could become a write-down information channel when a process of clearance X adds branches to an upgraded directory with classification X, causing the number of directory pages to grow. The "quota used" number in the superior directory would reflect this change and could thus be seen by a process whose clearance was lower than X. However, deleting branches will not cause the size of a directory to decrease. (This is true, due to the current implementation of directories.) If this facet of the quota problem was not eliminated, its usefulness as a means of compromising information for the malicious user is still very limited since:

- It can only be used by a Trojan Horse or cooperating processes (See Section 3.5.5).

- A process can only influence the size of a directory in a secondary manner, such as by creating a new branch and checking to see if the directory is large enough.

- A process can write-down only 6 bits (1 BCD character) per directory (1 to 64 pages.) Using two upgraded directories in the same parent will not be much help since it would provide only 7 bits, due to the additive nature of the "quota used".

- To use this information channel for writing-down N characters (6*N bits) in parallel, a malicious user would require N directories of the lower classification, each with an upgraded directory, and a starting pool of at least 66*N unused pages of his quota.

- A directory cannot be decreased in length by a process. This can only be done by a long salvage after a system shutdown or by deleting the directory (a process with the clearance to add branches to an upgraded directory does not have the clearance to delete the directory).

- A process must delete all branches in the upgraded directory and synchronize (using another directory) with a process of lower clearance to have the upgraded directory deleted and recreated, before another 6 bits of information can be passed. Otherwise, a record quota overflow will be reached rather quickly.

This information channel could be eliminated by charging all directory pages to its own quota. However, this involves a redesign of the entire quota mechanism and would impact the activation and deactivation of directories. Therefore, due

to the limited effectiveness of the information channel and the high cost of correction, no attempt will be made to close this write-down channel. However, for added assurance that it is not being used as a means for compromising data, the creation of upgraded directories and even the "get_quota" operation can be audited.

The Third Method

This problem is a result of the basic design of a hierarchical storage system which uses variable length branch entries and contains data of more than one classification. The space used by the branch of an upgraded directory is contained in a superior directory of lower classification. Hence, by adding ACL entries to an upgraded branch, one could affect the current length of a lower classified directory, which is in turn reflected in the "quota used" in the parent of that directory. This facet of the quota problem also requires the use of a Trojan Horse and is even more cumbersome than the others. It can only be eliminated by restricting upgraded branches to a fixed number of ACL entries. The changes described to close the second facet of the quota problem would not help this one. The solution of restricting ACL entries does not generalize properly for implementation and presents a very strange user interface. Until a correct long term solution can be designed, no attempt will be made to eliminate the last two facets of the quota problem.

3.7.3.2 The DTU and DTM Problem

Every branch contains two items known as the date-time used and the date-time modified. A process with clearance X can reference a segment with a classification lower than X causing its dtu to be updated. This updated dtu can then be observed by a process with a clearance lower than X and hence, write-down channel exists. In fact, whenever any segment is referenced, all of its superior directories must first be activated. Since activation is synonymous with use in the present system, the dtu's of all superior directories are updated whenever a segment is referenced. A similar problem is posed by dtm. The modification of a segment or directory causes the updating of dtm not only for that segment or directory, but for all superior directories as well. (This is done to aid the backup system in locating modified segments and directories without excessive searching.)

In order to eliminate the write-down channel caused by the upwards propagation of dtu and dtm, new interpretations will be given these two attributes with respect to directories.

Currently, dtu and dtm refer to the entire subtree of a directory. Instead, dtu and dtm will be made to refer to the directory itself. A new entry item called "date-time subtree modified" (dtsm) will be kept to maintain dumping efficiency. The dtsm, however, will be available only to the dumper process and not to ordinary users. (See Section 3.12)

In order to prevent writing down via dtu, it will be necessary to further alter the interpretation of dtu. Specifically, dtu will hold the time that a segment was last referenced by a process of the same "level" as the segment. In other words, the reading of a segment with classification X by a process with a clearance higher than X will be "transparent" as far as dtu is concerned. The same will also be true for directories. Notice that dtu will retain its present meaning for any segment which is referenced only by processes of the same "level." This change in meaning is acceptable, because dtu is primarily used in an interface where precision is not required. Dtu is primarily used to order the output of the list command and to delete all files not used in some period of time. Thus, a precise dtu is not essential.

Implementation of this new interpretation of dtu will be relatively simple. The global transparent usage switch (gtus) contained in each AST entry will be manipulated in a new fashion so as to properly control the setting of dtu. Whenever a segment is activated, the gtus will be turned off if the activating process has the same level and category set as the segment. Otherwise, the gtus will be turned on. Thereafter, any process which takes a segment fault on that segment will turn off the gtus if it has the same level and category set as the segment. The only exceptions to this rule will be special transparent system processes (e.g. the dumper and reloader) which will never turn off gtus. Whenever the branch of an active segment is updated, the dtu for the segment will be reset only if the gtus for the segment is off.

3.8 ACCESS TO I/O CHANNELS

3.8.1 Requirements

No user process should be able to directly attach any I/O device other than a terminal and then only if it has been specifically allocated to the process by a system process.

Each I/O device must be identified with a level/category.

Any process performing I/O on a device must only be able to perform the operations allowed by the fixed level property rules (i.e. only read from a lower "level" device, only write to a higher "level" device, and read/write to a device of the same "level").

The "level" of a device must be subject to change by the system operator.

The initial "level" of each device must be controlled by the System Security Officer.

Teletype channels must be identified with a maximum "level," so that a user can only create a process of a "level" equal to or below the maximum.

3.8.2 Design Considerations

One approach considered was removing current hcs_ entries for device attachment and using a new gate to restrict attachment of all devices other than terminals to system daemons only. This approach would require either a ring four write-around for hcs_ or else the modification of all ring four modules that reference hcs_ attachment primitives.

An additional consideration was to have the system control process manage teletype channels entirely. This concept went along with the previous approach, so that terminals could be handled in a slightly different manner and still be attached by user processes. Complete system control process management of teletypes was rejected because, with full system control process management of teletype channels, there are no ring 0 modules involved in the attachment decision, only in the actual attachment operation.

Another approach to managing I/O devices was to have separate device lists, one for system devices and one for user devices. The normal user attempting to attach any device would check only that list which applied to him. The configuration deck was suggested as a control for the memberships of these lists. This idea was also rejected.

The most promising suggestion involved adding a new ring 0 table, called the Peripheral Assignment Table (PAT), which would be referenced on each attachment operation, to provide a level/category check of the device that a process is attempting to attach. This idea was adopted as part of the Design Approach.

3.8.3 Design Approach

The Peripheral Assignment Table (PAT) concept will be integrated into existing ring 0 tables. Each device which could be attached to a process will be described in this table. The maximum mode, classification level and category set will be included in the entry for each device. If a process attempts to attach a device, the clearance of the process will be compared to the classification of the device to ensure that the process will not "write down" or "read up." The "write up" capability will be allowed only if the device is a "write only" device (e.g., a printer).

The use of the PAT, as described above, provides assurance that normal I/O operations will adhere to the fixed level property rules. It does not, however, prevent the possible exploitation of flaws in ring 0 I/O procedures. The exploitation of "bugs" contained in I/O procedures has been a traditional means of breaking the security of many computer systems. Therefore, until ring 0 I/O procedures can be certified correct, only trusted system processes will be permitted to directly attach any I/O devices other than terminals. This restriction will be enforced by moving attachment entries from hcs_ to a new gate accessible to system daemons only. An hcs_ write-around will be provided so that existing daemon software will not require modifications.

Any process requesting a tape drive to be attached must use the new ring one tape management software (TMS). The TMS will maintain a tape descriptor segment for each tape registered on the system. At attachment time the segment for the particular tape will be checked to find the requestor's "need to know" access and the classification of the tape. A message will be sent to the tape allocator process to assign the requesting process a drive of the same "level" as the tape. (Note: at this point, the ring one TMS is choosing the "level" of the drive based on the

classification of the tape descriptor segment.) When the mount request is sent to the operator, the drive classification will be specified (correctly) and the operator must verify that the tape requested has the same "level" as the drive. This can easily be done by color coding and plainly marking the correct classification on both reels and drives. The tape mount must be rejected (and the System Security Officer notified) if there is any discrepancy. (see Section 3.10 for more details on tape I/O). It must be noted that the primary control on tape security is the system operator. The TMS can only check the operator. If the operator makes a mistake or is "spoofed", the TMS cannot, in general, detect the error.

There must be a way to maintain operational procedure consistency and yet allow the system control process, running at the unclassified level (see Section 3.11), to read Top Secret backup tapes during reload. Operational consistency requires the Top Secret tapes to be mounted on a Top Secret tape drive. Therefore, a means will be provided for the system control process to bypass the fixed level property restrictions so it will be able to "read up" in a carefully controlled manner.

3.9 SYSTEM PROCESSES AND SYSTEM FUNCTIONS

Many system services such as logging in and logging out a user, printing a segment on the printer for a user, saving the contents of the disks on tape, restoring the contents of the disks from tape, restoring the contents of the disks when they have been damaged, retrieving a segment that has been lost, are performed by special processes, known as "system processes." Clearly, these processes need unusual power in order to be able to carry out their job, and, by their nature, cannot operate at any single clearance level without violating the fixed level property restrictions; however, they must never violate the fundamental security rules.

For example, some of these processes need the "read" and "write" capabilities on all segments in the system. Some need the "status" and "modify" capabilities on all directories in the system; some need to communicate back and forth with all processes in the system; some need to be able to attach any I/O channel. It is obvious that there exists no clearance which would give a system process the right to perform its job, and still adhere to the fixed level property requirements. However, for certification purposes, there is a very strong desire to assign a level and category to all processes in the system without exception. It is understood, of course, that system processes must not be bound by the fixed level property restrictions in order to perform certain tasks; therefore, the programs in these processes must "interpretively" enforce the fundamental security rules.

Use of interpretation rather than fixed level property rules by a system process, as part of normal system operation, will be called an "interpretive operation." Any interpretive operations should fall into one of the following classes:

a. Access to Segments: the retriever process and the system control process (when reloading) must be able to read and write segments of any classification, but only to copy properly classified information to and from tape. The I/O coordinator and also the system control process must be able to share message segments with user processes of any clearance.

b. Access to Directories: the system control process and the retriever process must be able to perform specific operations in any directory.

c. IPC: the I/O coordinator as well as the system control process must be able to receive "wakeups" from processes of any clearance, using the interprocess communication facility.

d. I/O Channels: the system control process must be able to attach an I/O channel of any classification (See Section 3.8).

Since it is desirable to minimize interpretive operations, the strategy for assigning a level to a system process is to select the one which causes the fewest interpretive operations.

An interpretive operation always involves a process, an object, and a time interval. For each interpretive operation which it performs, a system process must obtain an "exception permission." An exception permission can be represented by the triple (P,O,T) -- a process P is allowed to violate the fixed level property with respect to object O for time interval T. From the viewpoint of a given system process, each exception permission is represented by an object or set of objects and a time interval. For example, if the unclassified I/O coordinator needs to read a Top Secret message segment, the exception permission represented by

(all segments, lifetime of the process)

is sufficient to allow the interpretive operation to occur. A second permission,

(all message segments, lifetime of the process)

is more restrictive but still allows the operation. Finally, a third permission,

(all message segments, while the process is in ring 1)

is even more restrictive but still sufficient. Each exception permission has a smaller "exception envelope" than the preceding one. The second permission restricts the class of objects, whereas the third permission restricts the time interval as well. This example serves to motivate the notions of "object granularity" and "time granularity."

The second permission has a finer object granularity than the first, while the third permission has a finer time granularity than either of the first two. Granularity should be interpreted to be the scope or envelope of access granted to a system process for an interpretive operation.

For any interpretive operation, the finest granularity which still allows the operation is most desirable from the standpoint of the principle of least privilege. For the above example, the class of objects (which has only one member) represented by

> (the TS message segment for dprint queue 3 to
> the printer in room 405)

may well have the finest sufficient object granularity.

Two general approaches to maraging the use of exception permissions have been considered during the design analysis. These two approaches, called the "privileged function" approach and the "privileged process" approach, are described below.

The privileged function approach is one which permits the finest possible time and object granularity to be enforced. Essentially, this approach provides a special ring 0 primitive to perform each different interpretive operation. Access to these privileged functions is restricted to specific system processes by use of ring 0 gates having appropriate access control lists. Under this scheme, object granularity can be made as subtle as one desires. Also, time granularity can be tightly controlled. If an interpretive operation is performed entirely within ring 0, then the call into ring 0 and the corresponding return delimit the time interval of the exception permission. It is not only the absolute size of the time interval which is significant, but also the fact that control never exits the trusted ring 0 domain during the interval. Hopefully, this will reduce the effort needed to certify outer ring procedures which perform interpretive operations. The privileged function approach also provides a very natural and simple means for auditing interpretive operations.

The privileged function approach is not without its disadvantages and limitations from the viewpoint of impact on current implementation. The use of restricted gates tends to tie procedures to processes. Currently in Multics, system processes use many of the same library procedures as do other processes. If, however, system processes were required to employ special gates to perform privileged, but otherwise common operations (e.g. deleting a segment), then special versions of many library procedures would be needed. The daemon software itself would require numerous

modifications to convert calls to standard library procedures and standard ring 0 gates to calls to special library procedures and special ring 0 gates. And, of course, the new ring 0 privileged functions would have to be provided. Therefore, from an implementation standpoint, the privileged function approach is unattractive. Also, it should be recognized that this approach is not appropriate for all types of interpretive operations. For example, asynchronous events, such as the receipt of an ipc wakeup, cannot be handled in this manner. In many cases, the time interval of an exception permission cannot be tightly controlled. Consider, for example, the use of privileged functions to initiate segments or to attach I/O channels. Although the granting of these privileges can be restricted to ring 0, the subsequent use of these privileges cannot be so restricted. Hence, while it may not be difficult to locate the intervals within a program in which an exception permission is in use, it will be necessary to trace all possible side-effects. System auditors must ensure that a system process is memoryless with respect to each fixed level property exception. This will, in general, require full examination of every program which performs interpretive operations. Hence, a substantial certification effort will still be required for outer ring daemon programs.

The privileged process approach to handling interpretive operations is one which attempts to minimize implementation difficulty. In its simplest form, this approach merely requires a per-process switch to indicate whether or not a process has "system privileges." This switch (presumably stored in the pds) would be interpreted by those ring 0 modules responsible for access computation. Essentially, fixed level property access checking would be effectively disabled for all processes having system privileges. Clearly, this scheme requires comparatively little effort to implement. All that is necessary is a mechanism to initialize the privilege switch and modifications to suspend fixed level property access checking for processes having the switch turned on. Unfortunately, this approach pays little heed to the principle of least privilege. Also, this approach has the disadvantage that fixed level property exceptions occurring within a program will not be explicit in the code, but rather implicit in the fact that the executing process has system privileges. Thus, the task of certification seems more difficult as compared to the privileged function approach.

The basic privileged process approach could, of course, be greatly elaborated. Object granularity could be enhanced by use of multiple switches, each corresponding to a different class of objects. Also, time granularity could be enhanced by setting and resetting these switches frequently. Taken

to the limit, this scheme begins to resemble the privileged function approach. A switch, or set of switches, could be turned on before each standard ring 0 call, and then reset upon return. However, the finer the granularity, the more difficult the implementation; hence, the principal advantage of the privileged process approach is lost.

It is expected that some hybrid of the two approaches described above will be adopted in order to obtain a practical compromise between ease of validation and ease of implementation. The specific nature of the hybrid approach will depend upon design details to be considered during implementation.

3.10 I/O DAEMON CONTROL IN A SECURE ENVIRONMENT

3.10.1 Requirements

The primary requirement of the Air Force Data Services Center is that no user of the Multics system be able to directly control any external I/O device (other than his own terminal). Therefore, each I/O device must be controlled by a system process to provide the needed I/O capabilities. The devices that will be controlled by system processes will be the card reader, the card punch, central site printers, remote printers and tape drives.

For each line printer, an operator (other than the central site operator) will always be in attendance. This operator will be the primary "controller" of the line printer. The detailed requirements for operating local and remote line printers are as follows:

1. During operational hours, if the line printer is powered on and the system is running, the line printer should be kept busy as much as possible.

2. If the current queue being processed by a line printer is exhausted, another queue should get serviced automatically (within operational constraints). Separate queues will be kept for each cevice. For a given device, the queue 1 requests for any level should be processed before the queue 2 requests, etc.

3. There must be an accountability form terminal associated with each line printer (local or remote). Nothing will be printed on the line printer until the controlling process has attached the terminal by specific action on the part of the printer operator. During printer operations, there will be one accountability form produced for each copy of each segment printed (one per banner).

4. It must be possible for a printer operator to request a sample accountability form to be printed on the terminal to verify proper alignment of the forms.

5. It is required that both the accountability form terminal and the line printer be able to detect an "out of paper" condition and signal this condition to the process controlling the device.

6. It must be possible for the printer operator to start and stop operation of the line printer.

7. The printer operator must also be able to restart or reprint requests that are either in mid-execution or that have been processed but have not been processed correctly.

8. The amount of communication necessary between the printer operator and the central site operator must be kept to a minimum.

9. The banner for all printed output must identify the classification of the highest level of data that can be contained in the printout.

10. At the user's request, page headers and footers must be supplied on each page of printed output which will indicate the classification level of the segment from which the printed information was obtained. The header and footer labels will be optional, however the default will be to print labels. If desired, the user can replace the segment classification with an arbitrary string.

11. The current "header" and "destination" options will be retained for distribution point information only.

12. The accountability form will be filled in with all pertinent information relative to the request that it describes.

A new capability must be supplied to allow a system process to perform tape I/O based on user requests. The basic requirements for handling tape I/O are as follows:

1. Only system processes will be able to directly attach tapes.

2. Normal users will be able to place a tape read/write request in a queue for a system process to perform the actual information transfer.

3. When the tape data is online, the user will have to reference the data as a segment or multi-segment file.

4. Commands must be provided to allow users to make tape requests.

5. Tapes must only be mounted on physical drives of the same "level" as the tape.

Modifications must be made to the present card input scheme. The basic requirements for card input are as follows:

1. Only system processes will be able to directly attach the card reader.

2. The operational staff must not be burdened with the longterm storage and handling of a large volume of card decks.

3. The owner of a card deck will be responsible for identifying the classification of the deck at the time it is submitted to the operations staff for input.

4. A card deck submitted for input will be read into an online segment having the same classification as the deck.

The standard Multics card punching capabilities, which allows queued punch requests and user specified punch code, must be enhanced to identify the classification of the data being punched. The amount of card punch usage is anticipated to be low enough that system produced accountability forms are not required. A combination of administrative procedures and system software should be used to provide a secure method of distributing classified card decks.

3.10.2 Design Considerations

The message segment management design outlined in Section 3.5 forces the design away from the current Multics queueing strategy. For each device type supported we must provide separate queues for each classification level supported by the system. However, unclassified only degenerates to the current Multics strategy.

The design alternative of having one device driver for each permissible "level" for each device type was rejected due to the high overhead required in maintaining several "idle" driver processes and in having the I/O coordinator multiplex I/O devices and accountability form terminals between driver processes.

3.10.3 Design Approach

The approach for providing external I/O capabilities is essentially that of the Multics standard product, i.e. an I/O coordinator process and one driver process per device. For each device type supported by the system, there will be one message segment queue per "level" or, if desired, several queues per "level" having different priority ratings. The I/O coordinator must have the ability to access these queues at all "levels" and to communicate via IPC with driver processes at all "levels." The driver processes will obey the standard fixed level property rules concerning attachment of I/O devices and segment references.

I/O Coordinator

There is no "level" at which the I/O coordinator can operate strictly within the fixed level property rules. Therefore, it will operate at the lowest possible "level" with the special privileges needed. This choice offers the advantage of not requiring special IPC privileges for the driver processes with which the I/O Coordinator communicates. The I/O coordinator will have the following characteristics in the two-level security environment:

1. There will be multiple queues, specifically one per level per device class per priority.

2. The I/O coordinator will allocate tasks to various driver processes where each task is defined as a request of a single user.

3. The I/O coordinator will be responsible for making the decision of where to send an individual task, (i.e. to the appropriate device driver process at the correct "level"). The decision will be based in part on the minimum expected device level for a given class of device. This will, for example, allow the I/O coordinator to allocate all tasks for a remote line printer to a driver process at "level" n, if the remote printer is never to be classified below "level" n. At the AFDSC central site, where printers will be operated at both Secret and Top Secret, the minimum expected "level" decision criterion will prevent requests from Secret users and below being sent to a Top Secret device, so that there will be a minimum of over-classification at distribution time. The operator will be able to reconfigure the queues by changing the minimum expected "level" for a device class.

4. The I/O coordinator will not make decisions as to which device drivers to create. This will be done routinely by the system operator manually logging in the correct driver at the correct "level." The operator will also be responsible for reclassifying devices when necessary.

5. The I/O coordinator will have to perform interpretive security operations to be able to read and delete requests from each message segment queue at each "level" in the system. Also, the I/O coordinator must perform interprocess communication with driver processes at various levels.

6. A temporary history file will be recorded on a per driver basis for restarting requests, which have abnormally terminated or which were sent to a printer that had no paper.

7. The I/O Coordinator will be responsible for deleting segments when requested by a user. This task cannot be performed by the driver processes since, in order to allow for restarting, a segment cannot be deleted until some specified length of time after printing. Hence, the I/O Coordinator must bypass the fixed level property restrictions in order to delete branches from directories of all classifications.

8. Part of the optional data supplied by a user will be an event channel and process ID which can be used for user notification at the completion of his request, assuming that the process is still active at the time the request is processed.

9. The devices that will be controlled through the I/O coordinator and driver processes will be the card reader, the card punch, central printers, remote printers, and tape drives. There will be one driver process for each individual device.

Line Printers

Both local and remote line printers will be handled by printer driver processes. Printer driver processes will be operated with the following constraints:

1. The level of the driver will be equal to the level of the device. The level of the device will be used in determining the banner classification name for the printed output.

This process will not be able to access data in segments of a higher level.

2. The driver process will be passed requests generated from various "level" user processes as decided by the I/O coordinator.

3. The driver will add optional header and footer labels on each page of output indicating the level of the segment being printed. This will be explained in more detail later.

4. The printer driver process will be responsible for interpreting the "need to know" access of the requestor from the access control list of the segment that is being printed (The I/O coordinator will interpret the user's access for deleting, when requested.)

5. The driver process will require an accountability form terminal to be attached. At no time will the driver process attach its printer before the attachment of the terminal. If the terminal is inoperative, the printer is also assumed to be inoperative.

6. The driver process will be modified to prepare accountability forms.

7. There will be a sequence number associated with each banner sheet to help operations burst the printer output. Since this number will be generated by a driver process at request processing time, it will be unknown to the user. Therefore, it cannot be used as a claim check to pick up printed output.

8. Driver processes will accept commands from the accountability form terminal. These commands will be:

start	start printing requests
stop	stop at next request
abort	stop immediately
sample	print sample form

When the printer operator types "sample" on the terminal, the driver process will produce one sample accountability form to verify alignment of the paper. However, it will not start producing output until the operator enters the start command.

9. The driver process will prepare an accounting file to charge each user for the use of the printer.

Printer output will have a banner, optional page labels, and an accountability form to help identify the classification of the printed information. The banner will have an added field of large block lettering to indicate that the printed output "may contain <level>" where <level> is the classification level (without the category set) of the device on which the information was printed. The mnemonic used in the banner must be no longer than thirteen characters. (The same mnemonics will be used throughout the system.) The classification on the banner cannot be controlled by the user and will be the same as that indicated on the accountability form. In addition to printing the classification level in large block letters, the full classification, including categories, will be printed in standard-size letters.

The header and destination options to the dprint command will still operate as in the standard Multics system. However, the information supplied in this manner must not be used to determine classification of output. Rather, the information should be considered as user delegated "need to know" authorization to be used to help in the distribution of output.

Header and footer page labels may be placed on each page of printed output by use of a new dprint option. (The default will be no page labels.) The optional label will contain the classification of the segment from which the information was obtained. Alternatively, a user may request that an arbitrary string (less than 132 characters) be used in place of the segment classification by using another new option to the dprint command. Header and footer page labels will be centered across the page. It should be recognized that the use of page labels will reduce the number of text lines per page, and hence, may upset the page alignment of formatted output.

Tapes

Tapes will be handled as part of the general I/O scheme mentioned above. The "level" of a tape driver process will always equal the "level" of the requests which it handles. A tape driver process will be permitted read/write access to tapes having the same "level" and read-only access to tapes of a lower "level." Write-only access to higher level tapes will not be permitted since there is no apparent value in such a capability.

The user interface to the tape I/O mechanism will permit the user to request that a tape be read into a segment or that a

segment be written to tape. The user may optionally request notification (by means of an IPC wakeup) of the completion of a tape read operation. The user need not specify "standard" or "non-standard" tape format since this information will be available in the tape descriptor segment. Tape drivers will operate as follows:

1. The "level" of the tape driver process will be the same as the "level" of the requestor. However, only Secret and Top Secret processes will be used at AFDSC.

2. The "level" of the tape drive that will be chosen to satisfy a given tape request will be the same as the "level" of the tape as indicated by the tape descriptor segment. (The operator, however, is the primary control that the "level" of the drive matches the "level" of the reel.)

3. If the level of the driver process is greater than the "level" of the tape drive, the attachment will be read only.

4. Tape driver processes will operate within the fixed level property restrictions. Therefore, any segments created while reading tapes will have the same "level" as the driver process.

5. The access mode given to the requestor for a read request will be the minimum of the requested mode of access and the effective mode of access for that user to the tape descriptor segment.

6. On a read tape request, the information will be stored in a multi-segment file in a tape pool directory of the correct level using the tape number as the segment name (unless another pathname was specified by the user).

7. Storage management of the tape pool directories will be a problem. A tape image segment or multi-segment file (which can occupy thousands of pages of online storage) must remain in its tape pool directory long enough to be processed by the user. The required retention time will, of course, vary from one tape segment to the next. In order to allow the user to aid in storage management, a "delete" option will be provided for the tape write request. If specified, this option will inform the tape driver process to delete a segment after writing it to tape. As a further aid in storage management, it may also be desirable to give users modify permission in an inner ring to the tape pool directories. A command could then be provided which deleted a tape segment at the user's request while operating in an inner ring. This would ensure that a

user could only delete his own tape segments, and could properly handle the case of shared tape segments. Periodically, it will be necessary to delete from the tape pool directories those segments which have exceeded a specified age limit.

8. The "Multics standard tape" information stored in the tape descriptor segment will be used to identify which device interface module the driver process will use to read/write the tape. This provides a means of automatically handling both Multics standard format tapes and non-standard format tapes through the same user interface.

9. A tape write request will write whole tapes only. A tape read request may read a whole tape or else may specify a portion of a tape by supplying two end-of-file marks at which to start and stop reading. Individual records will not be read or replaced on a tape.

10. The user will be able to specify the pathname to be used for the read/write operation if he wants to use a different segment than would be provided in the tape pool directory.

    Note: The user must have enough quota to hold an entire tape if he wants to read a tape using a specified pathname.

Card Input

Card input will be handled much the same as in the present system. A user will submit his card deck to the operations staff along with a special control card specifying a userid. The deck will then be read into a segment created in a card pool directory, and the specified userid will be added to the access control list of the segment. There will actually be one card pool directory per "level." The owner of a deck will be responsible for identifying the classification of his deck and thus the appropriate card pool directory. Unlike the present scheme, no link to the card image segment will be created for the user. This eliminates a vulnerability of the present card input mechanism whereby a user could cause a link to be placed anywhere in the hierarchy. Instead, the user will be given a sequence number by which to identify the card image segment created for his deck. When logged in, the user will employ a new command which takes the sequence number as an argument,

locates the associated card image segment, copies it into a new segment named by the user, and then deletes the card image segment. This new command will operate in an inner ring. Users will have no access to card image segments or card pool directories in ring 4. Periodically, it will be necessary to delete from the card pool directories those segments which have not been copied and deleted within a reasonable period of time.

Card Output

Card output presents a new problem in identifying the classification of the information punched on the deck. Printed output is initially in one piece and each page can be labeled. Card decks, however, are not connected and cannot be labeled by the system except for deck header and trailer cards. Therefore, it is easy for card decks to get mixed with other cards of different classifications unless a new procedure is adopted.

The obvious solution is to use cards of different colors for the different deck classifications. This solution carries with it some operational problems which must be mentioned.

First, for this system to be effective, a given card color must always be used to identify the same classification. This is needed to ensure correct handling of the decks by the distribution staff and operations personnel. Therefore, if the supply of cards for a given color is exhausted, all card output for that classification must be suspended.

Second, a card deck of a given color is difficult to declassify since the meaning of the color must be preserved. Therefore, the downgrading of a card deck must be done by repunching the entire deck on cards of a different color and destroying the original. This operation must be administratively forbidden except under carefully controlled conditions and only when approved by the System Security Officer.

Third, to avoid the problems of over-classification, the punch must be handling requests of only one classification during any period of time. This means that operator intervention is necessary every time a request queue of a different "level" is to be serviced.

The Multics punch driver process will be modified to support this mode of operation by the following software capabilities and administrative procedures:

1. The punch driver process will operate within the fixed level property restrictions for access to segments and I/O channels.

2. To prevent over-classification of punched output, the driver process should operate at the "level" of the requesting process rather than at a higher "level." Also the "level" of the card punch should be the same as the "level" of the driver process. This will ensure that the color of the card deck will indicate the classification which corresponds to the clearance of the requestor. The I/O coordinator will help to separate the data of different request "levels" through the "minimum expected level" decision mechanism described above. Only if the operational burden of downgrading a portion of the decks punched is acceptable, should the "minimum level" of the punch be set higher than unclassified.

3. The I/O coordinator will inform the operator when the queue of requests for the current device driver is empty, to allow him to reclassify the device for operation at a new "level."

4. An operator will change the operating "level" of a punch driver by:

   logging out the driver process;

   reclassifying the punch to the new "level";

   changing the color of the card supply;

   logging in a punch driver of the correct "level."

   The driver process will inform the I/O coordinator of its clearance which will be used in routing user requests. However, the security of the punched output is totally dependent upon the correct card color being loaded by the operator.

5. Accountability forms for the card decks will have to be prepared manually. The normal terminal output of the driver can be used to separate the decks to ensure a one to one correspondence between accountability forms and card decks.

6. Users should be discouraged, administratively, from requesting a dpunch of a segment which has a classification which is lower than the clearance of his process. This would result in implicitly upgrading the information, resulting in overclassification.

7. The information provided by the terminal output of the driver process will be stored in a segment to provide an online audit of completed punch requests.

3.11 SYSTEM CONTROL PROCESS

3.11.1 Description

The system control process performs the tasks of system initialization, answering service and system control.

In its function as the system initialization process, it reads a system bootstrap tape and creates the Multics environment. If necessary, the system control process is used to reload the file hierarchy from backup tapes.

In its function as the answering service process, it "listens" to all teletype channels. When a terminal powers-on, it sends an interrupt to the system control process. The answering service then prints a greeting, and validates the login or dial command. In the case of a valid login command, the answering service creates a process in the name of the user, allocates the console I/O channel to the process, and sends the process a wakeup. The answering service also handles requests from the user's process for new_proc and logout, and coordinates requests for table updates from the System Administrator and Project Administrators.

In its function as the system control process, it records accounting information, validates requests for I/O devices (tapes, etc.), controls the consoleless daemons, and provides an operator command interface.

3.11.2 Requirements

It is a requirement that these functions continue to work, without substantial implementation modifications.

It is a requirement that the system control process violate the fixed level property as little as possible.

3.11.3 Chosen Approach

To minimize the need for special access and the necessity to rewrite code, the system control process will run at the unclassified level. In this way, all System Administration segments (e.g. user registration and accounting) will be unclassified. Thus, System Administration and Project Administration will be unclassified for nearly all functions and will require no violations of the fixed level property.

IPC (interprocess communication) will be modified to provide the necessary communication paths between the system control process and user processes. IPC will have a privileged entry to set a flag which will allow the system control process to receive messages from any "level" process, despite the fact that it is unclassified. By normal access rules it will always be able to send IPC messages to any process. (see Section 3.5)

In communicating with other processes, the system control process will be able to use specified message segments of any "level." (see Section 3.9)

The system control process, in its function as the system initializer, will initialize the ring 0 tables used to validate all attachments of I/O channels. (See Section 3.8.)

As part of its function as the system control process, it will execute operator commands for reclassifying I/O channels, handling tapes, etc.

See Section 3.3 for an explanation of absentee and login validation procedures.

See Section 3.13 for an explanation of the role of the System Control Process in reloading.

3.12 OTHER SYSTEM PROCESSES

3.12.1 The Backup and Dumper Daemons

Two system daemons, namely "Backup" and "Dumper," are employed to perform file system backup. These two daemons scan the hierarchy and copy to tape "eligible" files and directories. The eligibility of a file or directory for backup dumping depends upon the dumping mode. Incremental dumps, performed by the backup daemon, dump files and directories which have been modified since they were last incrementally dumped. Complete dumps, performed (less frequently) by the dumper daemon, dump all files and directories.

In the past, two separate daemons were needed in order to run incremental and complete dumps concurrently. However, a multiple login feature is now available which permits a user (or daemon) to be logged in several times concurrently with the same principal identifier. Hence, it is feasible to have only a single daemon for backup purposes. Therefore, Dumper.SysDaemon will be discarded in order to minimize the number of system daemons ard to simplify the access requirements for file backup.

The backup daemon will run with the highest clearance level so that it may read all files and directories. This implies, of course, that backup tapes and dump maps will, as desired, have the highest classification. The backup daemon, being a trusted process, will be permitted to directly attach tapes.

The backup daemon must set the date-time dumped (dtd) for all segments and directories. Currently, modify permission on a directory is needed to set dtd for branches contained within the directory. This implies that the backup daemon would need the ability to modify directories at all levels. This problem is really a manifestation of a more basic problem. Intuitively, it makes little sense that a user is forced to give the backup daemon modify permission to directories. The function of backup is essentially "read only" in nature. Therefore, read access to a segment will be sufficient to set dtd for that segment.

The date-time dumped (dtd), date-time modified (dtm), date-time used (dtu), and date-time entry modified (dtem) segment attributes will no longer be subject to explicit modification by users. Currently, these date-times are writeable via hcs_ and hence are not trustworthy. Therefore, hcs_ entries which set these date-times will be removed.

Certain changes to the dumper program (used by the backup daemon) will be required. First, the new level/category information must be backed up and hence must be added to the dump record format. Second, a new hphcs_ call must be provided to permit the backup daemon to set dtd. And third, a new branch attribute called the date-time subtree modified (see Section 3.7.5) must be introduced to guide incremental dumping.

The backup daemon will not violate the fixed level property rules in any manner.

3.12.2 The Retriever Daemon

The retriever daemon is used to recover selected files and directories from backup tapes at the user's request. In order to read backup tapes, the retriever must run with the highest clearance.

The retriever will require certain special privileges. In order to restore files and directories to the hierarchy, the retriever must be able to create branches of all classifications and to modify segments and directories of all classifications.

Certain modifications to the retriever program will be required. New ring 0 calls must be inserted to properly restore the classifications of segments and directories. Also, a new hphcs_ primitive will be provided to allow the retriever to restore the various date-times (since these will no longer be writeable via hcs_ as described above).

It is possible, although very unlikely, that an attempt could be made to retrieve a branch into a directory of a higher classification in violation of the non-decreasing classification rule of the hierarchy. This could only occur if a directory were created, deleted, and then recreated at a higher classification. (This sequence could also be simulated simply by swapping directory names.) Ring 0 will refuse to set the classification of a branch lower than the classification of its containing directory. Hence, an attempt to retrieve a branch into a directory of higher classification will implicitly reclassify the branch at the level of the directory. If a user wishes to avoid such

reclassification, he can rename or delete the existing directory, or else can retrieve the branch into a different directory (as described below).

It is also possible, but unlikely, that an attempt could be made to retrieve a segment into a directory of lower classification. This could only occur if a directory were created, deleted, and then recreated at a lower classification. Due to the quota problem (see Section 3.7.4), segments are required to have the same classification as their containing directory. Therefore, ring 0 will refuse to set the classification of a segment branch higher than that of its containing directory. Since the retriever cannot be permitted to declassify a segment, a request to retrieve a segment into a directory of lower classification must be rejected. A user can avoid this problem by renaming or deleting the existing directory or by retrieving into a different directory.

The SSO must develop a plan to administer the submission and validation of retrieval requests. Clearly, users cannot be permitted to directly inspect dump maps. This responsibility should probably be given to the SSO or his assistant. Retrieval requests can be submitted in person so that the requestor's identity can be validatec. Once the requestor's identity is known, some set of rules must be applied to determine the legitimacy of the request. Some alternatives include:

1. A user can retrieve anything under his home directory. A Project Administrator can retrieve anything under his project directory. A System Administrator can retrieve anything.

2. A user must have write access to the segment if it exists online. Otherwise, he must have modify permission for the nearest superior directory which exists online. These checks can be made by the SSO or his assistant. (Note that under this scheme, granting access to a segment implies granting access to the entire backup history of the segment. This should not be much of a problem, however, since segments are rarely "reused" for different purposes.)

Once a user's right to retrieve a segment or directory has been established, he can then retrieve that segment or directory into any existing directory in the hierarchy for which he has append permission. In most cases, a segment or directory will be restored to its original position within the hierarchy. In some cases, however, a user may request that a segment or directory be placed in a new position within the hierarchy. This is known as a "cross-directory" retrieval.

It may also be desirable to accept retrieval requests from remote locations. No formal mechanism currently exists for this purpose. In current practice, retrieval requests are sometimes accepted over the telephone. It should be noted that retrievals cannot be used in any manner whatsoever to declassify information. Hence, telephone-requested retrievals can be performed without fear of such a security breach. However, sabotage is possible by simply overwriting online segments with backup copies. Also, need to know access can be compromised by a cross-directory retrieval. Therefore, positive user identification should at least be required for all cross-directory retrievals, as well as for all retrievals outside of >udd.

3.12.3 The Repair and Ring_1_Repair Daemons

Two daemons, namely "Repair" and "Ring_1_Repair," are used to perform emergency fixes to the system. The Ring_1_Repair daemon runs in ring one in order to handle special ring one problems, e.g. the installation of a ring one gate. Both daemons require essentially unlimited access to the system via phcs_ and hphcs_. The repair daemons should run at system high, since they have access to all information in the system. They may have to "write down" information on occasion.

The passwords for these daemons should be known only to the SSO and should be changed after each logout. At his discretion, the SSO will make the passwords available to system programmers and other persons needing to use the repair daemons. It may be desirable for the system to actually require a password change for these daemons (performed by the SSO) between each logout and next login.

3.12.4 The Metering Daemon

The Metering daemon is used to generate system performance graphs and other system meters. For this purpose, phcs_ access is required. The daemon probably should run system high, because the metering information may be an information channel.

3.12.5 The Print_Dump Daemon

The Print_Dump daemon is sometimes employed to print BOS dumps (see Section 3.13.1) during normal Multics operation. A BOS dump may reside either on tape or in a special area of online storage known as the DUMP partition. At system initialization time, the initializer copies dumps from the

DUMP partition into the Multics hierarchy. These BOS dump segments, as well as BOS dump tapes, will be classified system high and hence, the Print_Dump daemon must run with system high clearance. This daemon, being a trusted system process, will be permitted to directly attach tapes. In current practice, the Print_Dump daemon may also directly attach a printer. In the security system, however, it is desired that all printed output be identified by a security banner. Therefore, direct printer attachment will not be permitted. Instead, formatted dump segments will be dprinted.

3.13 CRASH RECOVERY

3.13.1 BOS

BOS (Bootload Operating System) is a collection of programs used to perform a number of basic functions such as loading a Multics system tape, assisting in Multics shutdown, and dumping the Multics machine image (usually following a system crash). BOS also plays a significant role in file system backup and recovery operations. Periodically, Multics is shut down so that BOS may perform a "SAVE." A SAVE essentially copies all of online storage onto tape, thus establishing a checkpoint for use in file system recovery. In the event of online storage loss, BOS is called upon to perform a "RESTOR," i.e. the loading of online storage from SAVE tapes.

BOS will have no knowledge of Multics security controls. Operational control of BOS is sufficient to ensure security.

BOS dumps of the Multics machine image may contain information of all classifications and hence will be treated as Top Secret. BOS itself will provide neither security banners nor page labels for printed output. To do so would add unwanted complexity to BOS, and would require that specific classification names, e.g. "Top Secret," be included directly in BOS programs. Since such names are intended to be installation parameters, a different version of BOS would be required for every installation.

BOS dumps may be immediately directed to a printer, or else may be saved on tape or disk for later printing. In the former case, it is recommended that special paper be used to indicate the classification of the printed output, e.g. paper having a "Top Secret" water mark. If BOS dumps are printed during normal Multics operation, banners and page labels can be added at that time.

3.13.2 The Salvager

The salvager is a group of programs designed to detect, report, and correct wherever possible any inconsistencies in the Multics directory hierarchy. The salvager runs within a special version of the Multics operating system and utilizes

a separate partition of online storage. The salvager is employed following either a normal Multics shutdown or an emergency shutdown instigated by a system crash.

The salvager will be knowledgeable of security controls as they apply to the file system. The only kinds of security-related inconsistencies which can be detected by the salvager are violations of the non-decreasing classification rule of the hierarchy. Unfortunately, however, such inconsistencies cannot be automatically corrected. If an unclassified directory is discovered below a Secret directory, it does not seem warranted to delete the unclassified directory. It seems more reasonable perhaps to upgrade the directory and its inferiors where necessary since this cannot compromise security. However, this strategy may produce absurd results if, for example, >udd became erroneously classified Secret due to a system crash. Therefore, the salvager will mark a branch "out of service" whenever it fails to comply with security regulations. The pathnames of such branches will be reported to the operator. Explicit action by the SSO will be required after the system has been restarted to place these branches back in service.

The running of the salvager will be enforced by the system. Currently, when Multics is bootloaded without prior salvaging, a warning message is printed for the operator. Instead of just a warning, system initialization will actually be aborted.

There exist four different salvaging modes known as fast, active, regular, and long. A fast salvage merely flushes the paging device. In current practice, a fast salvage is usually performed after a successful emergency shutdown. When shutdown succeeds in recovering the file system device configuration table (FSDCT) from core, but fails to deactivate all active segments, an active salvage is sometimes performed. An active salvage examines all directories which could not be deactivated. If a shutdown attempt fails to recover the FSDCT from core, then a regular salvage is performed. Only a regular salvage will examine all directories and completely rebuild the FSDCT. Hence, only a regular salvage is guaranteed to detect all possible reused addresses, i.e. pages claimed by more than one segment. To avoid possible security violations, such pages are zeroed by the salvager. A long salvage is basically the same as a regular salvage except that it rebuilds all directories (not just inconsistent directories) for the purpose of directory compaction. It is recommended that regular or long salvaging always be performed so as to ensure file system consistency. (For the present MIT hierarchy, a regular salvage requires about ten minutes to run. Therefore, the time saved by use of the fast or active salvage modes is negligible. However, it is expected that

the time to perform a regular salvage will increase approximately linearly with the number of branches in the hierarchy.)

As with BOS, the salvager will provide neither security banners nor page labels for printed output. Special Top Secret paper can be used as suggested for BOS output.

3.13.3 Reloading

Following a system failure which causes extensive file system damage, it is necessary to recover the former contents of online storage from SAVE tapes and backup tapes. This recovery operation is known as a RESTOR/reload. The first step in the recovery procedure is to use BOS to perform a RESTOR of the latest SAVE. Next, Multics is bootloaded and backup tapes produced subsequent to the latest SAVE are reloaded in chronological order. Thus, the hierarchy is restored to its state at the time of the latest incremental dump.

The reloading of the file system from backup tapes is presently performed by the Initializer. The reason for choosing the Initializer is because other daemons cannot be logged in until the answering service begins operation. The answering service, in turn, cannot be started until all of its data bases have been reloaded.

When performing reloading, the Initializer will require certain special privileges. First, as an unclassified process, it must be permitted to read Top Secret backup tapes. Second, it must be capable of creating branches at all levels and writing at all levels. But as with the retriever, the Initializer is forbidden to violate the increasing classification rule of the hierarchy.

The reloader and the retriever programs share many of the same modules. Hence, the program modifications and the security considerations discussed in Section 3.12.2 apply to reloading as well as retrieving. It should be emphasized again that reloading, like retrieving, will never declassify information.

There exists another type of reload known as a "cold reload" which is not generally used as a method of crash recovery, but is sometimes used for special purposes such as directory reformatting. To facilitate major operations of this type, a complete dump is usually performed immediately before a cold reload. A complete dump is usually divided into sections, one of which contains all system files. These system files are reloaded first by the Initializer. Next, the answering service is started so that other system

daemons can be logged in to perform the remainder of reloading. The retriever daemon can be used for this purpose since it will have the necessary privileges.

3.14 OPERATOR INTERFACE

3.14.1 New Procedures and Responsibilities

Relatively few changes to the Multics operator interface are anticipated. The operators will be given the new responsibility of reassigning device classifications. Also, tape handling will be somewhat different. Tape drives and tape reels will be identified by color-coded classification labels. Each registered tape reel will have an associated three-letter authentication code to be typed by the operator at tape mount time for verification purposes.

3.14.2 Security-Related Messages

Security-related messages directed to the operator will explicitly specify violations, warnings, etc. so that appropriate operator action can be taken. Such messages will be distinguished by some convention, e.g. preceding asterisks. Also, the audible alarm on the operator's console will be used to alert the operator whenever his attention is required.

3.15 ADMINISTRATIVE CONTROL

3.15.1 Requirements

The functions of the System Administrator (SA) and System Security Officer (SSO) must be as distinct as possible. The SA must not be able to downgrade segments, nor assign classifications to new users, nor change the classification of existing users. The SSO must not be required to register new users nor perform accounting.

3.15.2 Design Considerations

The primary consideration is that the authority of the SA and SSO be clearly delineated. In this way, the SA will not be able to perform functions which are the responsibility of the SSO, and the SSO will not be burdened by the routine tasks of the SA. A secondary consideration is that the tools for use by the SSO should be simple and easy to use, and should follow normal Multics command conventions.

3.15.3 Chosen Approach

The SA will:

1. register all users;
2. perform system accounting functions;
3. perform default project administration.

In general, the tasks of the SA will remain unchanged in the new system.

The SSO will:

1. assign clearances to persons and projects, and assign classifications to terminals and I/O devices;
2. assign the mnemonics for levels and categories;
3. perform the downgrade functions on segments and directories;

4. be responsible for the physical security of all on-site and remote equipment, including the integrity of interconnecting cables;

5. be able to force a given user (or all users) to change his password;

6. receive and review all security audit data;

7. approve retrieval requests (see Section 3.12);

8. fix security-related inconsistencies detected by the salvager (see Section 3.13);

9. set the security audit flags for various personid's and projectid's (See Section 3.16.4).

The special commands (e.g. downgrade) used by the SSO will contain calls to auditing procedures to record their usage. It is also suggested that the console script of the SSO be retained as further auditing information.

Those privileged functions which must be restricted to the SSO alone will be implemented via a separate gate segment. In this way, the ACL on the gate segment can effectively be used to give access to the SSO while denying it to other users. The user ring functions (commands for inspecting the audit data, and setting the clearances of persons and projects) which are restricted to the SSO will similarly be protected by their ACL.

The tables which are shared between the SSO and SA are updated only by the system control process, and the updating tools will be modified to permit only the SSO to set the per-person clearances and audit flags in the PNT and the per-project clearances and audit flags in the SAT. In this way, the existing functions of the SA will not be affected, and the SSO will assume control of all security-related functions.

Several new tables will be the exclusive responsibility of the SSO, including the Peripheral Control Table specifying the I/O channel classifications and the terminal answerback codes.

3.16 SYSTEM AUDIT

3.16.1 Requirements

The system audit functions should provide a history of normal and abnormal system use, or operation, to permit regular security review of system activity (per DoD 5200.28-M). System events to be included in the audit data are:

1. each access to a classified file and the nature of the access (per DoD 5200.28-M);
2. each login and logout;
3. each unsuccessful login attempt and reason for rejection;
4. each rejected access to information due to security restrictions and each illegal attempted use (fault) of access permission;
5. all system faults which could indicate attempts to subvert the system or to exploit hardware failures;
6. all security related actions of the System Security Officer or the System Administrator;
7. each time a process awards itself extra privileges;
8. all completed requests for printed or punched output (not terminal output);
9. all tape mount requests for user tapes.

Where possible, the recording of audit data must have the capability of being turned off on a per user or per system basis. The subverter process, for example, must be known to the audit programs so that its known violations can, if desired, be omitted from the audit data. Data reduction programs must be provided to prepare summaries of audit data for inspection.

3.16.2 Design Considerations

Audit data segments must be writeable by many processes, hence, there must be a locking strategy provided with assurance that the process locking the data will unlock it before it loses eligibility. These segments must either fall outside the security rules, or there must be a data segment(s) for each level/category combination used on the system.

Ring zero auditing must be done only when there is no directory locked by the subject process to avoid deadlocking problems.

The feasibility of storing exponentially smoothed data on the interval between certain events will be examined (e.g. average period of illegal opcode faults, average period of initiate rejection due to security) after more design details are known and an assessment of performance impact can be made.

3.16.3 Design Approach

Each successful login is recorded on the system control console output, as well as in the online log kept by the answering service. This log also records each unsuccessful login attempt and the reason for rejection. The mechanism which records information in this log will be modified to ensure that the following data will be recorded for each unsuccessful login:

login line as entered by user
hardware channel of the terminal
answerback code of the terminal
maximum level of the terminal
the reason why the user was rejected
date and time

In addition, if the person's clearance is less than the maximum "level" of the terminal, a "breach of physical security" message will be sent to the operator. Also, if the number of bad passwords for a given personid is greater than the system maximum, an "attempted breach of security" message will be sent to the operator and recorded in the log. This count will be reset on the next successful login of that person.

All special commands provided for the System Security Officer to maintain security control will provide audit of their use. This data will be protected by the ring mechanism. However, the data can only be assumed to be complete if the security related function and its audit is performed in a lower ring than the System Security Officer

would log into. Otherwise, a person logging in as the System Security Officer could write a program which would perform the same security related functions without using the the audit interface. The details of which security functions must be performed entirely within the user ring will be described when the implementation details are better understood.

The granting of special access privileges to any process will be audited by ring 0.

Any rejected attempt to add a segment to a user's address space, due to security rules, will be audited by ring 0. (Shared data and locking problems will occur here).

All access violation and illegal opcode faults will be audited by ring 0. The data recorded for each fault audited will include:

pathname and offset of object denied access
type of violation (fault)
"level" of object
user's effective access mode to the object
"level" of process and current ring
pathname of executing procedure
user ID of process
date and time

(Shared data and locking problems will occur here).

The classified segment audit data will include:

user ID of process
pathname of the segment
"level" of the segment
user's effective access mode to the segment
date and time

This capability may introduce significant performance degradation in the system and will generate a large amount of audit data (shared data and locking problems occur here).

The problems of shared data and locking are primarily a problem associated with rings other than ring 0. The audit data areas must be writable by all processes if the information is to be complete. This cannot be done in the outer rings without allowing a user process to violate the fixed level property. Even if this was allowed, a process can lose its eligibility to use a processor while executing in an outer ring with a lock set which could cause other processes to wait for the locking process to be rescheduled.

Therefore, all auditing will be handled by ring 0 procedures since this is the only ring in which all processes can write in common data areas, regardless of clearance, without explicitly violating the fixed level property and since processes executing in ring 0 are guaranteed to complete all operations which must be performed while a lock is set.

The mechanism to be used for the ring 0 auditing will be an enhanced version of the system error auditing procedure. This has the advantage of providing a common interface for all system auditing functions. The audit data will be stored in a special disk partition and periodically copied into segments in the the Multics storage system by the system control process. The error type labels on each audit entry will be used to separate the security related entries from other system errors.

A ring 0 entry will be provided to allow administrative and security related procedures to record their actions as needed. Access to this gate should be provided for all users, but limited to rings of higher privilege than the normal user ring to avoid a potential source of sabotage through flooding the audit data with irrelevant entries.

The log of printed and/or punched output will be the file of accountability forms and an online copy of the information printed on the driver control console. User terminal output will not be logged. The system control console output and the system log data will provide the needed audit data for important system events not recorded elsewhere.

The allocator process that handles tape drives will provide a log of all tape requests, including denied requests.

3.16.4 Audit Selectivity

All processes will be treated the same by the audit system. The ring zero portion of the audit system will check the pds of a process to determine whether a given event should be included in the audit data. This will provide a wide range of selectivity to the audit system.

The audit flags in the pds will be established at process creation time. Another data field will be added to each entry in the PNT and SAT which will describe the events to be audited for each personid and projectid. At process creation time, the system control process will turn on the pds audit flags if the corresponding flag appears for the personid or projectid of the user.

Only the SSO will be able to turn off these flags in the PNT and SAT. The default value will be "on" for each new person or project registered by the System Administrator.

The events which will be identified by separate audit flags will include the following:

access to classified segments,

security related file system errors,

awarding of special access privileges,

illegal opcode faults,

access mode related access violation faults,

ring related access violation faults,

audit calls from outer rings,

other events identified during implementation.

It is recommended that the audit flags normally be turned off for the AFOSC supplied subverter process, since it will only add noise to the audit data. However, on occasion, it may be desirable to audit the subverter process as an aid in testing the audit system itself.

3.16.5 Audit Data Reduction

A simple data reduction and output formatting program will be provided to prepare audit data for inspection. For each class of audit data, the program will recognize keywords corresponding to fields in the audit data, such as "segment name," "userid," "error code," etc. The user of the data reduction program (presumably the SSO) will supply a list of keywords and corresponding data items. For example, if the user specifies "error code: no access," all entries in the indicated audit data file will be printed for which the error code field specifies "no access." A limited capability for the use of "AND," "OR," and "NOT" Boolean operations will be supported to enhance selectivity. The file_output command can be used to direct output to a segment.

3.17 CONTROL AND AUDIT OF SYSTEM CHANGES

3.17.1 Requirements

Security configuration control ensures that all changes to the operating system are accounted for and verifies that these changes do not impare the security of the system.

Procedures must be established to control and audit system changes. Software tools must be provided to assist in the audit. All security sensitive modules should be identified in each release. Source and object code have been provided for the initial release. Source and object code must be provided for all revisions along with a listing of all modules modified and the reason for each modification within each module.

3.17.2 New Release Material

For each new release, will contain at least:

A Multics system tape (MST);

Machine readable source code of all modules changed from Multics, BOS, Salvager, and DATANET 355 systems;

BOS and Salvager object tape if the code has been changed;

DATANET 355 object code if changed;

Object code of all compilers, assemblers, and PL/I operators used to generate the changed modules;

List of all modules changed with the reason for the change.

3.17.3 Tools

Procedures will be supplied to assist in comparison and auditing of system changes at source and object level. An ASCII comparison procedure will be supplied to aid in noting changes made to source code. A procedure which makes a

5.0 PREPARATION FOR DELIVERY

This section does not apply to this report.

6.0 NOTES

This section is for Administrative Information Only - Not Contractually Binding.

6.1 Removable Media

During the Design Analysis, the security requirements for integrating removable media storage into the virtual memory was discussed. The term "demountable segments" has been used in this section to differentiate removable media containing portions of the Multics storage system from removable media associated with I/O directed from a Multics process.

The recommendations resulting from the Design Analysis discussions are included in this section because they are not a direct part of the implementation of security controls. However, the following recommendations will be used as guidelines by the project which is designing the demountable segment capability for Multics.

6.1.1 Recommendations

1. The demountable segment capability must allow the basic Multics access controls to be extended to removable storage media. Disk packs are the primary type of media addressed, as the value of tapes in this role is not operationally clear at this time.

2. Each physical disk pack must be identified as part of the system, such that it should be impossible for it to be used by any process for direct I/O.

3. There must be a unique machine readable header on each physical unit. No disk pack should be usable for demountable segments until the header has been initialized by the system. This header should identify the highest classification of information ever stored on the disk pack.