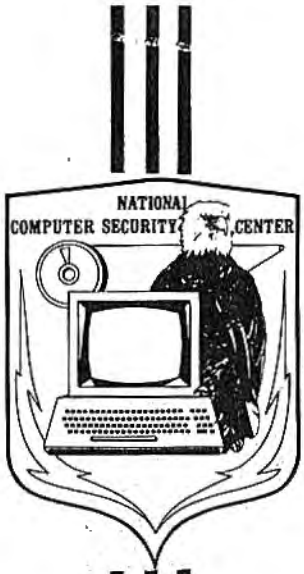

NATIONAL COMPUTER SECURITY CENTER

FINAL EVALUATION REPORT OF HONEYWELL MULTICS

MR11.0

1 June 1986

Distribution Restrictions
On Inside Cover

FINAL EVALUATION REPORT

HONEYWELL INFORMATION SYSTEMS

MULTICS MR11.0

NATIONAL
COMPUTER SECURITY CENTER

9800 Savage Road
Fort George G. Meade
Maryland 20755-6000

June 1, 1986

CSC-EPL-85/003
Library No. S227,783

CSC-EPL-85/003
Library No. S227,783

FOREWORD

This publication, the Final Evaluation Report Honeywell Information Systems, Multics MR11.0, is being issued by the National Computer Security Center under the authority of and in accordance with DoD Directive 5125.1, "Computer Security Evaluation Center". The purpose of this report is to document the results of the formal evaluation of Honeywell's Multics MR11.0 operating system. The requirements stated in this report are taken from DEPARTMENT OF DEFENSE TRUSTED COMPUTER SYSTEM EVALUATION CRITERIA (CSC-STD-001-83) dated 15 August 1983.

Approved:

June 1, 1986

Eliot Sohmer
Chief, Standards and Products
National Computer Security Center

ACKNOWLEDGEMENTS

Team Members

Team members included the following individuals, who were provided by the following organizations:

	The Aerospace Corporation Los Angeles, California
Deborah D. Downs Cornelius J. Haley Grace Hammonds David J. Lanenga Maria M. Pozzo W. Olin Sibert Eric J. Swenson Grant M. Wagner	National Computer Security Center Fort Meade, Maryland First Information Systems Group The Pentagon The MITRE Corporation Bedford, Massachusetts Oxford Systems, Inc. Arlington, Massachusetts

Further Acknowledgements

Technical support was also provided by F. Patrick Clark, Mindy D. Makuta, Sammy Migues, and Jay Steinmetz.

Acknowledgement is given to the following individuals for their contributions to this document: Walter S. Bond, Ann Marie Claybrook, Virgil D. Gligor, Jaisook Landauer, and Paul Woodie.

CONTENTS

This page intentionally left blank.

EXECUTIVE SUMMARY

The security protection provided by the Honeywell Multics MR11.0 operating system, with the B2-specific changes(1) applied, configured according to the most secure manner described in the Trusted Facility Manual, and running on the Honeywell Level 68/DPS or Honeywell DPS 8/70M multiprocessor has been evaluated by the National Computer Security Center (NCSC). The security features of Multics were evaluated against the requirements specified by the DoD Trusted Computer System Evaluation Criteria (the Criteria) dated 15 August 1983.

The NCSC evaluation team (see page iv, "Team Members") has determined that the highest class at which Multics satisfies all the specified requirements of the Criteria is class B2 and therefore, using the specified hardware (see page A-1, "Scope of Hardware Evaluation"), Multics MR11.0 (see page B-1, "Scope of Software Evaluation"), configured in the most secure manner described in the Trusted Facility Manual, has been assigned a class B2 rating.

A system that has been rated as being a class B2 system provides a TCB that is based on a clearly defined and documented formal security policy model that requires the discretionary and mandatory access control enforcement be extended to all subjects and objects in the ADP system. In addition, covert channels are addressed. The TCB must be carefully structured into protection-critical and non-protection-critical elements. The TCB interface is well-defined and the TCB design and implementation enable it to be subjected to more thorough testing and more complete review. Authentication mechanisms are strengthened, trusted facility management is provided in the form of support for system administrator and operator functions, and stringent configuration management controls are imposed. The system is relatively resistant to penetration.

A system that has been rated as being a B division system provides a Trusted Computing Base (TCB) that preserves the integrity of sensitivity labels and uses them to enforce a set of mandatory access control rules. The system developer has

---

(1) A set of "security critical fixes", distributed by Honeywell, must be applied to make the standard MR11.0 release fully B2-compliant, because of minor inconsistencies discovered during security testing.

provided the security policy model on which the TCB is based and furnished a specification of the TCB and evidence that the reference monitor concept has been implemented.

The Honeywell Multics system consists of the Multics operating system running on Honeywell Level 68/DPS or DPS 8/70M mainframes. These systems include Multics-specific hardware to support the Multics system architecture and protection mechanisms. A large Multics system may be configured with up to six processors and can support three hundred to four hundred users. Multics can be used in a wide variety of environments.

The Multics operating system is a general-purpose time-sharing system with strong security features. Multics has three basic security mechanisms. The hardware supported protection rings and segmentation provide tightly controlled separate domains of execution. The Access Isolation Mechanism (AIM) provides mandatory access control. The Access Control Lists (ACLs) provide discretionary access control.

INTRODUCTION

In December 1982, the National Computer Security Center (NCSC) began a formal product evaluation of Multics, a Honeywell Information Systems (HIS) product. The objective of this evaluation was to rate the Multics system against the DoD Trusted Computer System Evaluation Criteria (the Criteria), and to place its final rating on the Evaluated Products List (EPL). This report documents the results of that evaluation. This evaluation applies to Multics MR11.0 with the B2-specific "security critical fixes", available from Honeywell since February 1986.

Material for this report was gathered by the NCSC Multics evaluation team, through Multics documentation, interaction with system developers, and experience using Multics systems.

Evaluation Process Overview

The Department of Defense Computer Security Center was established in January 1981 to encourage the widespread availability of trusted computer systems for use by facilities processing classified or otherwise sensitive information. In August 1985 the Center was given responsibility for providing computer security guidance to the entire federal government and its name was changed to the National Computer Security Center. In order to assist in assessing the degree of trust one could place in a given computer system, the Criteria was published in August 1983. The Criteria establishes specific requirements that a computer system must meet in order to achieve a predefined level of trustworthiness. The Criteria levels are arranged hierarchically into four major divisions of protection, each with certain security-relevant characteristics. These divisions are in turn subdivided into classes. To determine the division and class at which all requirements are met by a system, the system must be evaluated against the Criteria by an NCSC evaluation team.

The NCSC performs evaluations of computer products in varying stages of development from initial design to those that are commercially available. Product evaluations consist of a developmental phase and a formal phase. All evaluations begin with the developmental phase. The primary thrust of the developmental phase is an in-depth examination of a manufacturer's design either for a new trusted product or for security enhancements to an existing product. Since the developmental phase is based on design documentation and information supplied by the industry source, it involves no

"hands on" use of the system. The developmental phase results in the production of an Initial Product Assessment Report (IPAR). The IPAR documents the evaluation team's understanding of the system based on the information presented by the vendor. Because the IPAR contains proprietary information, distribution is restricted to the vendor and the NCSC.

Products entering the formal phase must be complete security systems. In addition, the release being evaluated must not undergo any additional development. The formal phase is an analysis of the hardware and software components of a system, all system documentation, and a mapping of the security features and assurances to the Criteria. The analysis performed during the formal phase requires "hands on" testing (i.e., functional testing and, if applicable, penetration testing). The formal phase results in the production of a final report and an Evaluated Products List entry. The final report is a summary of the evaluation and includes the EPL rating, which indicates the final class at which the product successfully met all Criteria requirements in terms of both features and assurances. The final report and EPL entry are made public.

Document Organization

This report consists of five major sections and two appendices. Section 1 is an introduction. Section 2 provides an overview of the system hardware and software architecture. Section 3 provides a mapping between the requirements specified in the Criteria and the Multics features that fulfill those requirements. Section 4 details the testing procedures that provided evidence and assurance that Multics met the B2 requirements of the Criteria. Section 5 presents the evaluation team's impressions of the system. The appendices identify specific hardware and software components to which the evaluation applies.

SYSTEM OVERVIEW

This section begins with a brief description of the history of the Multics product. The remainder of the section describes the security-relevant architecture and mechanisms used in Multics. The information presented in this section was verified by the evaluation team through examination of documentation, interaction with system developers, system analysis, and experience using the Multics system.

Background and History

The Multics product results from a 20-year product development effort, and served as an example for many of today's operating systems. The Multics project started in 1965 as an effort by Massachusetts Institute of Technology (MIT), Bell Labs, General Electric, and the Air Force to produce a secure, flexible "computing utility". In late 1965, a set of six Multics design papers was presented at the 27th Fall Joint Computer Conference, describing the overall design. From 1965 to 1969, MIT (Project MAC), General Electric, and Bell Labs jointly developed the Multics hardware (the GE-645 processor) and software. By 1969, Multics was in general use at MIT running on this hardware.

In 1971, Honeywell Information Systems acquired the General Electric Computer Division, and with it, the Multics system, including new hardware under development at that time (which later became the Level 68 processor). In 1974, Multics became a Honeywell product, and the first commercial (non-MIT) software release (MR1.0) was produced.

The MR1.0 software was the first to run on the new hardware, then known as the H-6180 (and now as the Level 68). This was a great improvement over the GE-645, both by using more modern hardware technology and by having a hardware implementation of protection rings, the fundamental security and integrity mechanism in Multics. Prior to the H-6180 and MR1.0, this protection had been provided in software, and was considerably less reliable(1) and effecient.

---

(1) The penetration effort described in the Multics Security Analysis: Vulnerability Report (Karger, P. & Schell, R., ESD-TR-74-193, Vol. II, June 1974), for example, was conducted on the GE-645 hardware, and all the problems it identified were directly related to the lack of hardware ring protection. The study's conclusion was nonetheless that

From 1975 to the present, Honeywell has continued to develop the Multics software and hardware base. In 1976 (MR3.1), the Multics Mandatory Access Control mechanism, AIM, was first provided, and later that year (MR4.0), the New Storage System software was released, providing a crash-resistant file system and eliminating the significant object reuse problem. In 1980, MR8.0 provided the Resource Management package, extending the system's security model to include disk and tape volumes as well as files. Throughout the development history, security has remained of paramount importance, and both discretionary and mandatory security protection mechanisms were considered in the design of all new subsystems.

In 1981 (MR9.0), Honeywell released a new line of Multics hardware, the DPS-8M. It is software-compatible with the earlier Level 68 systems, and provides all the same security mechanisms. This evaluation covers the entire Multics hardware product line (see page A-1, "Scope of Hardware Evaluation").

The 1985 release of MR11.0 represents the latest stage in the Multics product line. It continues the emphasis on security and flexibility, a combination that currently serves the requirements of customer sites worldwide.

Multics System Architecture

At its inception, Multics embodied the most advanced operating system and software engineering principles of its time, and it has adhered to those principles throughout its 20-year lifetime. It is that continual practice of good software engineering that allows Multics, unlike most other systems of its era, to achieve the degree of architectural consistency required for a B2 evaluation rating.

The major contributor to architectural consistency in Multics is the use of virtual memory for all access to stored data. Rather than having a multiplicity of mechanisms for I/O, buffer allocation, memory management, etc., Multics references all data directly through memory reference instructions, and relies on the virtual memory (demand paging and segmentation) to make the information accessible transparently. Although it might

---

Multics was the most secure commercial system available, and that its security would be greatly enhanced by the new H-6180 hardware.

initially appear that using demand-paged virtual memory for all data access would be unacceptably inefficient, that cost is largely compensated for by the simpler system architecture resulting from a unified approach. Another compensation is the relative ease with which a single mechanism (the virtual memory) can be optimized compared to the difficulty of optimizing a colleotion of independent mechanisms.

The major factors in Multics' architectural consistency, both within the TCB and outside, involve common use of shared function. With the exception of some low-level TCB code, the entire system (and all user applications) uses the same virtual memory mechanism for data reference. Dynamic linking allows the easy sharing of subroutines, ensuring that a particular function is always performed the same way, by the same code, whenever it is needed. The programming languages share a common runtime environment and debugging tools, even allowing much TCB code to be debugged in a user environment with some simulation support.

Multics Hardware

This section describes the hardware components that make up a Multics system, and the types of protection mechanisms they provide. Only those components that implement part of the protection mechanisms are inoluded; individual peripherals (disks, tapes, printers, etc.) are not discussed, since they contain no security-relevant mechanisms. To satisfy requirements for system integrity, all hardware must operate correctly; the procedures and tests used to verify correct operation are discussed on page 81, "System Integrity".

Major Hardware Components

Multics has five major hardware components, not including peripheral devices, each of which is described separately in subsequent subsections. Multiple models of some components are available, but they differ only in performance and implementation technology, not in security-relevant features. Only a very small percentage of TCB code behaves differently for different models of hardware components, and the differences are not security-relevant. More variety is available in models of peripheral equipment, but since peripherals are not security-relevant, they are not discussed in this report.

CPU - Central Processing Unit

The CPU executes programs. The CPU provides all protection mechanisms for programs and data. A single system may contain up to a total of eight CPUs and Input/Output Multiplexers (IOMs) combined. Each CPU connects to a single port of each System Control Unit (SCU) in the system.

The two major models of CPU are the Level-68 and the DPS-8. The two models are very similar in hardware architecture and implementation, the major difference being the higher speed and the newer implementation technology of the DPS-8. The instruction sets are identical, except that the DPS-8 CPU also implements both an extended-range floating point ("hex mode") and a somewhat different set of privileged hardware control instructions.

Both the Level-68 and DPS-8 CPUs have several submodels, differing only in performance (different processor speeds and cache architecture). For purposes of covert channel bandwidth estimates, the fastest available processors were assumed.

This evaluation covers all models of the Level-68 and DPS-8 CPU hardware. System security is not affected by the differences in hardware and software implementation.

SCU - System Control Unit and Memory

The SCU contains central memory and provides the communication path between CPUs and IOMs. A single system may contain up to four SCUs (eight if only Level-68 CPUs are used). The SCU contains no protection mechanisms of its own, since it is referenced only by the Trusted Computing Base (TCB). Each SCU is attached to every CPU and IOM in the system. An SCU can contain up to 4 million words of memory.

CPUs and IOMs communicate with an SCU via "ports". An SCU has eight ports, each of which is connected to one CPU or IOM. Hardware configuration switches determine which ports are active, and how much memory is available, ensuring the integrity of the connections. These switches may be set either by the operator or by the TCB.

In addition to housing system memory, the SCU provides the path for dispatching I/O interrupts, and inter-processor signals, called "connects". When an I/O operation is completed, or one

CPU wishes to interrupt another, an SCU is responsible for accepting that signal from the device (CPU or IOM) attached to one port and sending it to another.

IOM - Input/Output Multiplexer

The IOM provides the path between peripherals and central memory. Each IOM is connected to one port of each SCU in the system, and has connections ("channels") to all the peripherals it can control. The IOM implements comparatively sophisticated protection mechanisms for user-written I/O programs (see page 14, "IOM Protection Mechanisms").

The IOM controls peripherals. Each peripheral is connected through an IOM channel. A peripheral may be a single device (an operator's console, for instance), a controller for multiple devices (tapes and disks), or an independent processor (the Front-end Network Processor, FNP). Some devices (FNPs) are manipulated only by the TCB, some are available either through the TCB or directly (disks), and some are not used by the TCB at all (tapes). The TCB uses the hardware protection mechanisms to ensure that devices are used appropriately.

Multics treats the different models of IOM (see the list on page A-3, "Additional IOMs and IOM Enhancements") as completely equivalent, except in the lowest level hardware device driver programs. The IOM model differences are not user-visible.

MPC - Microprogrammed Controller

The MPC provides the interface between an IOM channel and one or more peripherals (such as a string of disks or tapes or a set of unit-record devices: printers, card readers, etc.). The MPC is not directly user-visible. It is an independent processor and executes a fixed microprogram that is loaded during system bootload. It can be reloaded during system operation for testing or recovery from hardware failure. The MPC processor has a very simple architecture: the microprograms are typically only 8K bytes long and are provided with the hardware.

The only protection mechanism provided by the MPC is to ensure that a single I/O program is permitted to reference only a particular device; the TCB ensures that every user-written program starts with a device assigned to the user, and by the MPC ensures that the I/O program references only that device.

There are several models each of disk, tape, and unit record (line printer, card reader, card punch) MPCs (listed on page A-3, "Disk MPCs and Options"); the differences between models are not significant to the user, although they are visible in some cases, since the I/O programs for tape and other user device I/O are user-provided, rather than being part of the TCB. Each MPC model requires different firmware (listed on page B-7, "MPC Firmware").

FNP - Front-End Network Processor

The FNP provides the physical and logical interface between an IOM channel and all serial communication channels. It is an independent processor and executes a fixed program that is loaded during system bootload or during system operation. The FNP is relatively sophisticated. The program ("core image") it runs must be individually configured for each system to include the protocol handlers for the serial protocols in use on that system. The FNP core image is approximately 64K bytes, written in FNP assembler. Each core image is bound together from a subset of about 20 different modules, which implement various serial protocols.

The only user-visible interface to the FNP is through the TCB. Unlike other user-controllable I/O devices, the user cannot write I/O programs to communicate with the FNP, but must make specific requests (read, write, control) through the TCB, and the TCB translates these requests into appropriate I/O programs.

All protection mechanisms relevant to the FNP are implemented in TCB software, either in the central system or in the FNP code itself. The I/O hardware protection mechanisms are used only as an integrity aid. The FNP software is completely isolated from user programs, and therefore reasonably safe, despite being written in FNP assembler language. It includes a variety of internal integrity checks on memory allocation and on correct operation in general. When any integrity check fails, the FNP crashes, and is automatically reloaded with a fresh version of the core image by the central system. The FNP-resident software is part of the TCB, but it implements no explicit security policy, because its instructions come from the central system TCB, which does implement the security policies for communications channels.

Only one model of FNP exists, with various performance enhancements and options (see page A-4, "Communication Processors and Options"), as well as channel interfaces to handle asynchronous communications and many types of synchronous protocols.

Hardware Protection Mechanisms

This section describes the user-visible protection mechanisms provided by the hardware components. The hardware also provides extensive internal protection mechanisms that protect against and detect malfunctions, but these are not user-visible, and not security-relevant except from an integrity standpoint.

Most user-visible hardware protection is provided by the CPU. These protection capabilities include: segment access control, which automatically enforces software-defined access modes on every machine reference; the "effective ring" mechanism, which ensures that more privileged procedures cannot be "spoofed" into referencing normally inaccessible data; the gate mechanism, which allows for very rapid, hardware-mediated transfers between different privilege levels; and privileged mode, which controls access to the hardware features providing protection.

The remaining hardware-provided protection affects user-supplied I/O programs. Multics allows a user to execute arbitrary I/O programs to control devices assigned to his process. The hardware mechanisms are used to prevent those I/O programs from affecting any devices or memory other than those assigned to the user's process.

Per-Segment Access Control

The segment is the fundamental unit of access control in Multics. A process has the same effective access to all data in a segment, but different processes may have different degrees of access to the same segment.

Access to a segment is defined by two values: the raw access modes and the ring brackets of the segment. The three-bit raw access mode allows for read, execute, and write permissions (see page 39, "Discretionary Access Controls"). The ring brackets determine what effective ring is required to exercise those rights (see page 10, "Effective Ring Number in Addressing").

Every segment has three ring numbers: R1, R2, and R3, and three access bits: R, E, and W. A process may write into the segment if the W bit is on and the effective ring is between zero and the R1 value, inclusive. A process may read the segment if the R bit is on and the effective ring is between zero and the R2 value, inclusive. The rules for instruction execution (the E bit, and R3) are described on page 11, "Gates and Ring Changing".

The hardware makes these checks at every access to the segment, since the effective ring may change in each instruction. If the access is not permitted, an exception is signalled to the supervisor, which can log the attempted access violation, and the exception is signalled to the user.

The hardware also checks other segment attributes at every reference, such as whether the segment is accessible at all (defined for the process), and whether the access is within the maximum size of the segment.

Effective Ring Number in Addressing

Most systems offer hardware access checks based only on the current state of execution: privilege level, ring, or protection key. This means that privileged system software must carefully validate every address supplied by less privileged programs to ensure that the data addressed can be accessed by the less privileged program. This software-based mechanism is a common source of protection errors, since it requires explicit action for every address.

Multics uses an ordered hierarchy of eight rings (0-7) for hardware protection, assigning progressively more privileged functions to lower-numbered rings (see page 17, "Software Ring Usage"). This protection is enforced by hardware, and is the heart of the reference monitor implementation.

Unlike most other systems, however, Multics bases access checks not on the current ring (privilege level), but on the history of the process that generated an address. The hardware automatically maintains a ring number with every address, and that ring number is set to the maximum possible ring (least privileged ring) that could have affected the value of the address. Thus, when a more privileged routine uses an address, the hardware can automatically check access with respect to the history of that address, not merely with respect to the current level of privilege.

This hardware mechanism is called the effective ring number. When any instruction references memory, as part of generating the final address, it generates an effective ring number, representing the privilege level of the reference. For each instruction, the effective ring number begins as the current ring of execution. As the instruction proceeds, the effective ring is compared with other ring numbers developed during the instruction, and it is increased to the maximum (least

privileged) value of all those ring numbers. This effective ring, not the ring of execution, is used in determining whether an access is permissible.

In addition to generating the effective ring during every instruction, the CPU also maintains an effective ring with every pointer register, and stores an effective ring (as part of the stored pointer, not an invisible tag) whenever a pointer register is saved in memory. These ring numbers indicate the effective privilege level of that address and are used in the ring maximization. When an address is loaded into a pointer register, the effective ring of the pointer register is set to the effective ring generated during the instruction; and when the pointer is stored, that ring number is saved with it. Whenever a pointer register is used to generate an instruction address, or whenever an indirection is performed through a stored pointer, the effective ring associated with the pointer register or stored pointer takes part in the maximization.

The other source of ring numbers in the maximization process is the ring brackets of the segment in which a stored pointer resides. If a segment can be written in ring two, then the effective ring of any pointer stored in that segment is two or greater; if the value stored within the pointer is zero or one, it is ignored. Thus, even if a program in ring four fabricates a pointer with a ring number of zero, it cannot store it except in segments writable from ring four, and therefore if it is passed to the ring zero supervisor, the effective ring number for any instructions using it will be at least four.

By taking the maximum of all these ring numbers, the hardware ensures that when it generates the final address, its effective ring will be as great as the maximum ring that could have affected the value of the address. Even if more privileged software stores that address in its own storage and uses it again later, the effective ring number will have been saved with it and will be used again.

Gates and Ring Changing

In order to execute instructions in a segment, the E bit must be on, and the effective ring must be between the R1 and R2 values inclusive (i.e., R1 <= eff ring <= R2). Transfers of control using ordinary transfer instructions must always generate an effective ring equal to the ring of execution. If they do not, an exception will be signalled, because only two special instructions are permitted to change the ring of execution.

The two special instructions are CALL6 and RTCD, which are used to perform the call and return operations respectively. They are simply transfer instructions (the stack discipline is implemented entirely in software). The CALL6 instruction can, when appropriate, decrease the ring of execution (increasing privilege) when performing an "inward call". The RTCD instruction can increase the ring of execution (decreasing privilege), when performing an "outward return". Outward calls and inward returns are not permitted.

An inward call is permitted only when the called segment has the "gate" attribute, which is specified by an additional access bit (G). If a segment has the gate attribute, and the E bit is set, a CALL6 instruction may transfer to it as long as the effective ring is between the R1 and R3 values inclusive. If the ring of execution is greater than the R2 value, it is decreased to R2 (increasing privilege). A gate segment has an additional attribute, the call limiter, which specifies the number of locations to which CALL6 transfers may be made. This allows the privileged code (such as the supervisor) to ensure that it is entered only at specific entry points.

To ensure an undisturbed environment for execution of the more privileged program, the CALL6 instruction takes an additional step: it selects a stack segment for the program to use. This is done by setting the stack base pointer to point to a per-ring stack, whose address is defined in a privileged hardware register. Once this is done, the program can determine all other information about its execution environment from the stack header and cannot be affected by any manipulations from the less privileged rings. When the CALL6 instruction does not change rings, the stack is not changed.

An outward return transfer via RTCD is permitted whenever the effective ring is between the R1 and R2 values for the target segment. Because this does not represent an increase in privilege, no restrictions are placed on where an RTCD may transfer. In practice, it must return to the location immediately following the CALL6 that transferred to the more privileged ring.

All calls and returns, regardless of whether they cause ring changes, use the CALL6 and RTCD instructions, so that a call to the supervisor is coded identically to a call to any other subroutine. To aid in handling unexpected conditions in more privileged rings, a hardware Ring Alarm Register (see page 51, "The Validation Level") is provided.

When an exception is signalled, the ring of execution is changed to zero, and control is transferred to a specific location in the supervisor. The supervisor (see page 20, "Fault Handling/Signaling") is responsible for saving context, processing the exception (possibly logging it, or signalling it to the user program), and continuing processing. The same is true for interrupts. Exceptions and interrupts cannot behave like subroutine calls because subroutine linkage is handled entirely in software.

Calls which would increase the effective ring (outward calls) and returns which would decrease the effective ring (inward returns) are not permitted, and generate an exception if attempted. Outward calls are not permitted because they represent a dependence by more privileged software on the actions of less privileged software; there is no way for program running in one ring to be confident that a program it calls in a less privileged ring will do what it wants (or even that it will ever return at all). Similarly, inward returns are not permitted because the more privileged program being returned to has no way to ensure that the less privileged program is returning to a valid location. Unlike inward calls, which are restricted by the call limiter to a known set of entrypoints, inward returns have no such restriction.

When specifically required, a program can effectively perform a "call" to a less privileged ring by simulating a valid outward return, although the less privileged ring can never return inward to the "caller". This capability is used only during process creation, since every process starts running in ring zero, and must be explicitly switched to its intended ring of execution after its ring zero databases are created.

A complete description of the rules for access checks, ring validation, and exception generation is given in the Multics Processor Manual.(1)

Privileged CPU Instructions

Certain critical instructions are privileged, meaning that they can be executed only in ring zero, and only if the P bit is set in the segment descriptor for the executing segment. Since

---

(1) DPS/Level 68 & DPS 8M Mutlics Processor Manual. Honeywell Information Systems, Inc., AL39-01B, February 1982.

modification of segment descriptors is not a privileged operation, and any ring zero program can do so, this provides only an additional separation of privilege within ring zero.

The most important privileged instructions are LDBR (Load Descriptor Base Register), CIOC (Connect I/O Channel), and SCU/RCU (Store Control Unit/Restore Control Unit). Most of the others manipulate internal registers in the CPU or SCU and are necessary to support page control, segment control, the scheduler, dynamic reconfiguration, and diagnostics. These other instructions have no important security aspects.

The LDBR instruction is used to change process address spaces. The Descriptor Base Register defines the current descriptor segment, and thus the entire address space, so that changing it changes the current process. The LDBR instruction is used only by the scheduler.

The CIOC instruction sends a "connect" to another CPU or to an IOM. It is used to start all I/O operations and to interrupt other CPUs.

The SCU and RCU instructions save and restore internal CPU state and are used in all fault and interrupt processing. When a fault (exception) or interrupt occurs, an SCU instruction is used to save the state of the CPU at the time of the fault, including at what precise stage of instruction execution the interruption occurred. The handler for the condition, in ring zero, determines the type of fault and performs the appropriate action (such as, for a page fault, initiating a page read, forcing the faulting process to wait for the page, and scheduling a new process). If the fault can be restarted (such as any I/O interrupt, or a page fault after the page becomes available), an RCU instruction is executed, and execution resumes at the precise point where it was interrupted

IOM Protection Mechanisms

The IOM is used to execute I/O programs that control I/O devices and transfer data between a device and memory. An I/O program consists of a series of Device Control Words (DCWs), of three types: the IDCW (Instruction), which specifies a particular I/O operation, the DDCW (Data), which specifies a memory address and transfer length, and the TDCW (Transfer), which transfers to another DCW at a different location. An I/O program is initiated by creating a set of control words in a known (protected) location, describing the I/O program, and sending a signal (a "connect") to the IOM by use of the privileged CIOC instruction.

The control words that describe the I/O program also specify a channel number, specifying a particular device or MPC. The channel number cannot be changed once the I/O program has begun execution.

An I/O program begins execution in "absolute " mode, which gives it access to all of system memory. Subsequent IDCWs, or even the control words used to start the I/O themselves, can specify restrictions on this addressing. The two types of restrictions are "rel mode", which restricts all subsequent memory accesses to a contiguous region of memory described by a base and bounds in the I/O program, and "paged mode", which restricts all subsequent memory accesses to a 256K word region described by a page table, and only to those pages marked "present" in the page table. Once either of these restricted modes is entered, neither changing back to absolute mode nor switching to the other restricted mode is possible, so all subsequent accesses are bound by the same restrictions.

When in a restricted mode, the entire range of addresses specified by a DDCW must lie within the addressing constraints, as must the target address of a TDCW. These addresses are zero-origin within the I/O program, relative to the base of the I/O buffer, and are automatically relocated (rel mode) or translated (paged mode) by the IOM. An I/O program stores status when it completes and may also store status at intermediate points during its execution. This status is stored either in a fixed location or as part of a queue within the addressing restrictions of the I/O program. Thus, storage of status, as well as data transfers, are protected.

See page 44, "Use of I/O Protection Mechanisms", for a discussion of the software that uses the IOM protection mechanisms and the MPC protection mechanisms described in the next subsection.

MPC Protection Mechanisms

Some I/O devices (for instance, all tape drives and disk drives) are attached to Multics through Microprogrammed Controllers (MPCs). For these devices, the object of any I/O operation is described both, by the IOM channel number, used to select a particular MPC, and the MPC device number, used to select a particular device attached to the MPC. The channel number is supplied when the I/O is initiated and cannot be changed by the I/O program (see page 14, "IOM Protection Mechanisms"). The device number, however, can be specified in every I/O command in an I/O program, and therefore is subject to manipulation by the user.

Changing the device number is prevented by the MPC firmware, which prohibits any change of device number once an I/O program has begun execution. Although the device number may appear in every I/O command, once a non-zero device number has appeared in one command, the MPC firmware ignores all subsequent specifications of device number. Device zero is special and is used to select the MPC itself, rather than an attached device, to allow for loading of firmware, reading error statistics, etc. If a device number of zero is specified in one command, a subsequent command can specify a non-zero device number, and select that particular device. However, because device zero is inaccessible to non-privileged users, this is not a problem, since a user-supplied I/O program must begin by specifying a non-zero device number.

Software Architecture

This section describes the overall software architecture of the Multics TCB: how it is structured and organized, and what functions are performed by its various components. Subsequent sections discuss the objects provided by the TCB, details of the implementation, and specific protection mechanisms.

The Multics software architecture is structured around the "rings of privilege" implemented by the hardware. Software is categorized by its "ring of execution": the level of privilege it has while executing. A given process actually runs in multiple rings; most code runs in ring four with calls to inner (lower-numbered) rings to invoke TCB functions. The Multics TCB consists of three major parts, each of which is discussed in a section below: the ring zero supervisor, the ring one administrative programs, and the privileged processes.

The ring structure has two major functions. The first is data integrity: data in a more privileged ring is protected against being modified by code running in a less privileged ring. For example, a forum meeting (an online meeting facility) is writable by all its participants, but only in ring two, and therefore is under the control of the forum program.

The other function of the ring structure is selective privilege: code running in a ring can specify which functions it will perform on behalf of code running in a less privileged ring. For

example, certain functions performed in ring zero may be invoked only(1) from code running in ring one, whereas others may be invoked from code running in rings one through seven.

Software in a more privileged ring has certain implicit hardware protections from interference or spoofing by data in less privileged rings, but it must also follow certain conventions in order to be secure. For details, see page 46, "Software Protection Architecture".

Some TCB functions are performed by dedicated processes which have special access and can invoke TCB functions in ring zero and ring one that are not accessible to other processes. These processes usually run in ring four, but since they do not need to share their data with other processes, the data are protected by ACL, unlike shared TCB data, which must be protected by rings since these data potentially can be manipulated by all processes. These dedicated processes are described on page 24, "Dedicated TCB Processes".

Software Ring Usage

The Multics rings are used as follows:

Ring zero
: The most privileged, "supervisor" ring. It contains the shared supervisor, which includes all the basic access control mechanisms, as well as other functions normally associated with operating system supervisors, such as the process scheduler, the file system, hardware device drivers, interrupt handlers, and interprocess communication. Most of the TCB runs in ring zero.

Ring one
: The "administrative" ring. Ring zero and ring one contain all portions of the TCB that can be invoked by arbitrary processes (some TCB functions are handled by entirely separate processes; see page 24, "Dedicated TCB Processes"). Software in ring one has the privilege of violating the mandatory flow control policies. This privilege is used to implement message segments and to share TCB data for the Resource Control Package (RCP) and logical volume management.

---

(1) This refers to external interfaces only. All functions performed in a particular ring may also be invoked by that ring, if appropriate.

Rings two and three
These are reserved for shared applications: ring two for those supplied by the vendor (such as forum), and ring three for those developed by a site. Because these applications are less privileged than the TCB, they are subject to all the TCB's access control policies, but because they are more privileged than user code, they can safely share data (subject to the TCB's rules) with more precise control than simple read/write permissions.

Ring four
Ring four is the standard "user" ring. Processes normally execute in ring four, unless otherwise specified by an administrator. Most of the privileged processes run in ring four, but have access to special TCB interfaces to perform their privileged functions.

Ring five
Ring five is used, at the discretion of project administrators (who have control over a group of other users), for less privileged users. In the same way that a site can use ring three to run an application shared by all users at the site, a project administrator can use ring four to run an application shared by all users on the project, and force all those users to be able to login in ring five only.

Rings six and seven
These rings are not generally used and are available for future expansion.

The execution environments for rings one through seven are essentially equivalent, except for the access and interfaces available. However, the execution environment in ring zero is different in several ways, primarily in that static storage is shared by all processes executing in ring zero, rather than being unique to a process. This allows the supervisor to function very efficiently, because a process running in ring zero can easily reference all data in ring zero. Programs in other rings must explicitly declare all data that is to be shared.

Ring Zero TCB Functions

The ring zero part of the TCB is the most sensitive portion of the TCB; it contains the underlying primitives used for all access decisions. The ring zero TCB performs many of the functions traditionally performed by the "operating system" in other systems: paging, the file system, the scheduler, etc.

The environment in ring zero is shared by all processes. Unlike the other rings, where the same segment may have a different segment number in each process that uses it, all processes have access to the same set of ring zero segments, which all have the same segment numbers in each process. Only five segments in ring zero are different in each process: the Descriptor Segment (DSEG), which defines the entire process address space; the Process Data Segment (PDS), which contains all miscellaneous per-process ring zero information; the Known Segment Table (KST), which describes all the segments addressable by the process outside of ring zero; the Processor Data Segment (PRDS), which contains per-processor software information (a process is assigned the appropriate PRDS by the scheduler when it is assigned to a CPU for execution); and the ring zero stack (stack_0), which is used for calls and returns within the ring zero supervisor.

The important ring zero subsystems are listed below, along with the specific security-relevant mechanisms they implement.

Page Control and Volume Management

Page control and volume management are responsible for multiplexing physical memory (paging) and managing the allocation and freeing of disk records, and their assignment to segments. The primary security-relevant function of page control (beyond merely operating correctly in mapping data into process address spaces) is satisfying the object reuse requirement (see page 53, "Object Reuse"). Other than monitoring potential covert channels, page control makes no access decisions of its own. It relies on segment control and directory control to have already made the access decision in order to allow page faults to occur. Whenever a record is allocated or freed, page control checks the object reuse assumptions about disk record allocation, in case an otherwise undetected hardware failure occurs during normal operation and damages the allocation data.

Segment Control

Segment control is responsible for maintaining a process address space and multiplexing segments among processes. It makes no explicit access decisions, but calls upon directory control to make them. Segment control does some monitoring of potential covert channels and validates the AIM labels of segments before allowing them to be used. Segment control is responsible for auditing successful access to segment contents.

Directory Control

Directory control is responsible for all user-requested operations on the file system. Unlike page control and segment control, which operate either to satisfy page and segment faults or in response to calls from directory control, directory control is mostly a user interface. Directory control makes all the basic access decisions for segments and segment-based objects (directories, message segments, etc.). It is also responsible for all auditing of access decisions and user-requested operations.

Address Space Management and Dynamic Linking

Address space management is responsible for making a segment, from some place in the file system, addressable by a process. It is invoked either by explicit user request ("initiate") or by the dynamic linker, which automatically resolves references between programs running outside ring zero. It relies on directory control to make access decisions, and updates the process DSEG and KST appropriately. The dynamic linker, although it runs in ring zero, does so only for implementation convenience, since the inter-procedure linkage information it manipulates resides outside ring zero.

Fault Handling/Signaling

Fault handling is responsible for locating the correct procedure in ring zero to handle an exception (such as a page fault or an overflow). If no specific action must be performed by ring zero, the fault is signalled outside ring zero. A user program may handle the fault, and request that it be restarted. In this case, ring zero must ensure that the machine conditions being restarted are valid for the process and do not represent any potential compromise. Ring zero fault handling is also responsible for auditing access violation faults.

Traffic Control

Traffic control, or the scheduler, is responsible for selecting processes to run on the various processors. Processes are scheduled according to administratively set tuning parameters governing the distribution of CPU time among users. The scheduler is also responsible for interprocess communication (IPC) and makes the access control decisions for IPC channels, auditing invalid access attempts.

I/O Interfacer (IOI)

IOI is responsible for processing user-supplied I/O programs, such as are used for tape I/O. IOI does not make any access decisions about devices and channels, except to allow a process to manipulate only those devices assigned to it. It

relies on RCP to have supplied that information correctly before it allows the user to perform any operations. IOI sets up the I/O hardware protection mechanisms (see page 14, "IOM Protection Mechanisms") and manages the memory for I/O buffers. The only auditing performed by IOI records invalid operation attempts.

Hardcore I/O Support

Hardcore I/O support is responsible for actually manipulating the I/O hardware, initiating I/O programs, and doing I/O required within the ring zero TCB. It includes the low-level I/O support, the disk I/O mechanisms used by page control and segment control, and the I/O support for the system console. It is also used by IOI to initiate the user-supplied I/O programs processed by IOI.

Multics Communications System (MCS)

MCS is responsible for managing all I/O to and from communication lines and all communications with the Front-end Network Processors (FNPs) to which those lines are attached. MCS makes no access decisions of its own beyond those required for "Software TTYs" (STYs) and those to ensure that a process operates only on communication channels that have been assigned to it. The decision to assign a channel is made by the Initializer.SysDaemon privileged process. Users may not communicate directly with the FNPs, but send buffers back and forth via MCS. MCS is responsible for managing all internal buffers and control of the FNPs. In addition to controlling hardware communications I/O, MCS provides STYs, a data transmission facility (like pipes) which appears to the user as two communication channels connected back to back. Because this connection is performed within the TCB, MCS must ensure that the two processes connected to opposite ends of a STY have equal security labels, so they can read and write data to each other. Other than monitoring invalid operations, no security-relevant auditing is performed by MCS.

Initialization and Shutdown

System initialization and shutdown is responsible for ensuring that the system starts and finishes operation in a secure state. The entire ring zero supervisor is rebuilt during each initialization (bootload) based on the contents of a Multics system tape, which is generated online from a description of the ring zero supervisor and the system libraries. It makes no explicit access decisions, but the process of initialization sets the ACLs on all gates into ring zero as specified in the description file. After a system failure, initialization is also responsible for auditing the presence

of inconsistencies in some TCB databases, and either correcting them automatically or referring them to the system administrator for manual correction.

Ring One TCB Functions

The ring one portion of the TCB is essentially a set of system-provided application programs that run in ring one, and therefore, have access to some services performed by ring zero that are not accessible outside ring one. Unlike ring zero, where the environment is very special, the ring one environment is much like that in ring four: all the paging, segmentation, and file system services are available, and the full runtime environment is available.

The major service provided to ring one is multi-class segments. Unlike ordinary segments, which may contain information at a single sensitivity level only, a multi-class segment may be referenced over a range of sensitivity levels, and contain data at all those levels. Multi-class segments may be created and referenced directly only from ring one, and the ring one TCB's function is to ensure that it always maintains the correct sensitivity labels on the individual objects contained within a multi-class segment. The minimum level in the range for a multi-class segment is the level of its containing directory. Depending upon how ring one chooses to specify it, the maximum level is either the maximum level allowed to the process creating it (not necessarily the current level) or "system_high" (the maximum possible level). Because the ring zero TCB does not enforce the mandatory flow policy for multi-class segments, the ring one TCB must perform this enforcement itself, and ensure that the information stored within a multi-class segment falls within that range.

Ring one uses multi-class segments both as containers of individually labelled objects (message segments, RCP registries), where the objects are accessible to users through ring one interfaces, and as a means of sharing per-system data with the ring one TCB running in other processes.

Some other ring zero interfaces are available only to ring one; these are used primarily to inform ring zero of an access decision made by ring one (such as RCP informing IOI about access to a device).

These are the major ring one subsystems:

Message Segments and Mailboxes
Message segments are a general mechanism for storing individually labelled objects within a single segment. System request queues and the mail system's mailboxes are the two main users of this mechanism. The ring one code interprets extended access modes instead of the usual read/write controls, and therefore, can allow operations that cannot be allowed directly on segments, such as "write-up". The message segment primitives audit their own access decisions, and ring zero does not audit the operations it performs on their behalf. They also audit message segment overflows, which are a potential covert channel event.

Resource Control Package (RCP)
RCP and RCP Resource Management(1) (RCPRM) control all user-accessible I/O devices and tape and disk volumes. The information about each RCP-controlled object is stored in the multi-class RCP registry segments and is manipulated only through ring one. RCP is responsible for multiplexing devices among processes as well as maintaining the control information about its objects. It is responsible for informing IOI that a process is allowed to use a particular device, possibly after being used (usually through an operator authentication command) to authenticate (verify the identity of) a requested volume. It audits the results of all its access decisions.

Master Directory Control and Logical Volume Management
These two subsystems control administration of logical volumes and creation of directories that use those logical volumes. Most of their functions are used only by administrators, but any user may potentially need to reference the master directory databases, if only to determine the name of a volume on which a particular segment resides. All of these subsystems' operations are audited.

Administrative Database Management
Certain administrative tables (the Person Name Table and Terminal Type Table) can be manipulated by multiple administrators; because they are shared and because changes in the tables must be audited, they are maintained in ring one.

---

(1) Resource Management is an RCP option that must be enabled for a site to run as a B2 system. Throughout this report, the term "RCP" refers to RCP with RCPRM enabled and fully in use.

Dedicated TCB Processes

Certain TCB functions are performed by dedicated processes, known as the "privileged processes". These processes have access to special TCB interfaces or resources that allow them to perform functions that cannot be performed by ordinary processes. The programs run by these processes are part of the TCB, even though they run in ring four;(1) although ordinary ring four processes can attempt to run the programs, they will fail because they do not have appropriate access.

The privileged processes are distinguished primarily by the gate segments to which they have access; the Multics Trusted Facility Manual describes in detail the gates, the processes, and what access is required for what functions. Many dedicated TCB processes also run with the project ID of SysDaemon, which, by default, gives them access to all directories and segments in the file system (although a user can remove this access in order to prevent access to specific files or directories). Finally, the Initializer process has special meaning to the ring zero supervisor, because it is the "system control" process.

The privileged processes do not adhere strongly to the principle of least privilege. Except for the Initializer, which has more privilege, and the volume backup processes, which adhere quite strictly and have no unnecessary privileges, all privileged processes have essentially the same access to gates and resources. However, this shared access is not a requirement for the functions being performed, and the current situation is due primarily to historical reasons.

Since the privileged processes have essentially unlimited access to the file system, those that process user requests must interpretively evaluate the access of the requesting user with respect to the requested operation. Decisions of this type are made by the Initializer, the I/O processes, and the Volume_Retriever. No interpretive access decisions are made by the other privileged processes.

---

(1) Actually, some privileged processes run in ring one, but treat the environment the same as ring four; that is, they do not make explicit use of any of the TCB interfaces available only from ring one. This is done primarily for hierarchy backup and retrieval, to allow those processes to back up data on behalf of all rings without using a special file access mechanism.

The dedicated TCB processes are:

Initializer.SysDaemon
This is the "system control" process. It is responsible for system initialization and shutdown as well as performing various utility functions for the TCB. It is responsible for creating and destroying all other processes and for enforcing process access controls. It is responsible for assignment of all communication channels, and logical volumes. It handles all input and output from all the daemon processes, directing daemon output to system consoles and logs. It processes all operator commands, also logging the operator input and command output. It maintains the system log files in which all TCB-generated messages are stored for use by the audit reduction tools. It handles certain commands sent by administrators and mediates all changes to the tables that define users, projects, resource definitions, and communication channels. When processing a request from a user, the Initializer must interpretively check access to the access control segment controlling the requested resource or facility.

Utility.SysDaemon
This is a utility process that is used primarily to perform functions that are logically part of the Initializer's responsibilities, but have been moved out to decrease its load. These include deleting any per-process data left over from previous bootload (which only happens after a crash); copying the image of a crash dump; periodically polling hardware (memory, MPCs, FNPs) for hardware errors; and monitoring quota usage in critical system directories. The site can add other functions to this list, such as real-time monitoring (see page 77, "Audit") of the audit logs for suspicious events.

IO.SysDaemon
The IO SysDaemon processes are responsible for running all system printers, and one exists for each active printer (or punch). By virtue of SysDaemon access, they have access to all user files and are responsible for ensuring that printer output is given the correct access labels; therefore before processing a file, the I/O daemon must interpretively verify the requesting user's access to it. Non-privileged I/O daemon processes may also be created (by running them on a project other than SysDaemon). These non-privileged I/O daemons are bound by the standard access control rules and therefore are only capable of operating single-level devices, but they still make the interpretive access checks, and are therefore part of the TCB.

Backup.SysDaemon, Dumper.SysDaemon, Retriever.SysDaemon
These are the hierarchy backup processes. They are responsible for writing and reading the system-controlled hierarchy backup tapes. By virtue of SysDaemon access, they can read and write files belonging to all users.

Volume_Dumper.Daemon, Volume_Reloader.Daemon, Volume_Retriever.Daemon
These are the volume backup processes. They are responsible for writing and reading the system-controlled backup tapes of individual disk volumes. This form of backup is more efficient than hierarchy backup, but does not allow for convenient recovery of anything but entire disk packs and individual files. Consequently, most sites use both forms, using hierarchy backup for long-term backup, and volume backup for short-term recovery from hardware failure. The volume backup mechanism does not require SysDaemon access because it uses a special interface for reading and writing data. The volume retriever automatically processes retrievals (of individual files), taking requests from a queue. It must evaluate, in an interpretative manner, the requesting user's access to the retrieved object and target location before processing the request.

Salvager.SysDaemon, Scavenger.SysDaemon, Repair.SysDaemon
These are the processes used to run the periodic maintenance operations required to recover from damage caused by crashes or disk failures, and to reclaim space in directories wasted due to fragmentation.

TCB-Protected Resources

This section describes the resources (objects) provided and protected by the TCB, the access modes a subject may have with respect to those objects, and the Multics implementation of Discretionary Access Control and Mandatory Access Control.

The Multics TCB provides and protects a variety of resources. All software-controlled resources are protected in some fashion. Most resources are protected by an explicitly associated Access Control List (ACL) and mandatory security label, which are stored either with the object itself (segments, directories) in TCB-maintained tables (RCP-controlled volumes and devices), or by explicit association of an access control segment with the object (communications channels). These resources are summarized below.

Some resources (system tables, metering data, system logs, system debugging information) are accessible only by privileged users and consequently are protected only by the ACL on a gate segment that allows access to the resource, or the ACL on the file containing the information itself; although they have mandatory security labels, these resources may contain information at multiple security levels that is not properly reflected by that label. They are not listed here because these resources are available only to privileged users, and the nature of the information and the privileges required are described fully in the Multics Trusted Facility Manual.

Objects Implemented by the TCB

This section lists all the TCB-protected resources, identifies the software that controls access to the resource, describes where their discretionary (ACL) and mandatory (label) access control attributes are stored, and describes how the security label of the object is set. Of course, if following the rules for setting the security label would result in a violation of the constraints on the level, the creation or assignment is not permitted.

Once set, the security label cannot be changed, except either by privilege or by returning the object to the system (unassignment for devices and communications channels, deletion for all other objects). Some of the rules for the value of the security label can be overridden by use of privilege. Since this capability is not relevant to non-privileged users, it is not discussed here; see the Multics Trusted Facility Manual for details.

Segments

Segments are containers for information and can also be used as access control objects by associating a specific segment (and its ACL) with another resource. They are managed by the "file system" in ring zero. The ACL is stored in the containing directory, the security label in the containing directory and Volume Table Of Contents Entry (VTOCE). The security level is set to the level of the containing directory, which must be equal to that of the process creating the segment. The security level must be within the range permissible for the logical volume on which the segment resides.

Directories

Directories are segments which contain segments or directories: the building blocks of the hierarchy. They are managed by the "file system" in ring zero. The ACL is stored in the containing directory, the security label in the containing directory and VTOCE. By default, the security level is set to that of the containing directory, which must be equal to that of the process doing the creation; however, the creating process can specify any higher (dominating) security level up to the maximum permitted for the user.

Message Segments

These special segments are used as containers for mail messages and queue entries which have their own distinct security labels. Three different types of message segments are provided: queue message segments, used primarily as queues for requests to a server process; mailbox message segments, used for mail and interactive messages between users; and user-message message segments, used for communication between the user processes and the Initializer process. The different types of message segments all implement the same access modes, or a subset (see page 34, "Message Segment Extended Access Modes"). Message segments are managed by segment control and directory control in ring zero and the message segment software in ring one, and may be manipulated directly (as segments) only within ring one. All user access to message segments is mediated by the ring one TCB. As for a normal segment, the ACL of a message segment is stored in the containing directory, and the security label for the entire segment in the directory and VTOCE. The security level is set to the maximum permitted for the user creating the message segment, and the message segment is allowed to contain messages of any level between that of the containing directory and the message segment itself.

Messages

Messages are the individual entries in a message segment. They are managed by the message segment software in ring one. They have no explicit ACLs of their own; rather, discretionary control depends only on the author of the message and the ACL on the entire message segment. Each message has an individual security label stored in the message header, and a single

message segment may contain messages with many different labels. The security level is set by default to that of the sending process, but can be increased (upgraded) either explicitly by the sending process, or automatically if a higher-level process receives the wakeup and the message at the time it is sent.

Non-System I/O Devices, Disk and Tape Volumes

Non-system I/O devices, disk volumes and tape volumes are RCP-controlled (Resource Control Package) I/O devices and volumes with no internal structure known to the TCB. They are managed in ring one by RCP. They are recorded in the RCP registries, which are segments managed by RCP in ring one and inaccessible outside of RCP. By default, RCP permits access only to the original owner of the resource. This can be changed by recording the pathname of an owner-specified segment in the registry; the ACL of that segment is used as the ACL for the resource.

The security label, actually a range, for these resources is stored in the registry entry. The security level range restricts use of the device to processes whose security level falls within the range. The resource range must fall within the range allowed for the resource type as defined in the Resource Type Definition Table, a system table writable only by the Initializer. When an RCP resource is acquired by a user, it receives a security label with equal upper and lower limits, both set to the current level of the acquiring process. For devices, this assignment is temporary. When the user releases the device, another user at a different level can acquire it. For volumes, the assignment is permanent. Once the volume is written, it retains the label it received until it is explicitly cleared by a security administrator.

Communication Channels

Communication channels are recorded in the Channel Definition Table (CDT), a system table writable only by the Initializer.SysDaemon process. All management of channels to other processes is performed by the Initializer. The ACL for a channel is kept on a segment of the channel's name in a system directory (>scl>rcp). The security label, actually a range, is stored in the CDT. As for RCP devices, the range restricts use of the channel to processes whose security level falls within the range. When a channel is assigned to a

process, the current security level for the channel is temporarily set to the security level of the requesting process.

System Disk Volumes (Logical Volumes)

Logical volumes are groups of disks usable by the "file system" (page control, segment control, directory control, etc.). They contain individually labeled segments and directories (as opposed to RCP-controlled disks, which have no internal structure known to the TCB). Logical volume registration is managed in ring one by the logical volume and master directory control software. This information may be updated only by a privileged user. The ACL for a logical volume is kept on a segment whose location is stored in the registration segment. The security label, actually a range, of a logical volume is stored in its registration segment, which is kept in a system directory (>lv), and also in the volume label of each disk in the logical volume. The range restricts the security levels of the segments that are created on that volume. The minimum and maximum values in the range are specified by the system administrator when the volume is initially defined to the system.

Processes (Interprocess Communication)

The only operation a non-privileged process may perform with respect to another process is communication. Interprocess communication is controlled in ring zero by traffic control (the process scheduler). Process creation is performed by the Initializer process. Processes have no explicit ACLs; however, to communicate with another process, a process must know the cryptographically protected signature of one of its IPC channels. This provides discretionary control, since that information will be unknown unless published by the owning process. The security label for a process is stored in the Active Process Table Entry for the process, as well as in various places in per-process storage. The label is derived at process creation time (see page 41, "Mandatory Access Controls") from the security level ranges stored in the administrative tables defining users (PNT), projects (SAT, PDT), and communications channels (CDT).

Object Access Modes

The following section describes the effective access modes a subject may have with respect to the various object types. A subject's "effective access" to an object is determined by the combination of the Discretionary Access Control (DAC), Mandatory Access Control (MAC), and ring mechanisms. It represents the set of operations a subject may perform on an object.

Discretionary access (see page 39, "Discretionary Access Controls") is determined from the Access Control List (ACL) associated with the object, and the set of modes in the ACL term matching the subject name. An ACL term may specify any combination of access modes for an object. In all cases the "null" (n) mode indicates the absence of all other modes.

Mandatory access (see page 41, "Mandatory Access Controls") is determined by the relationship (›=, meaning "dominates"; ‹=, meaning "dominated by"; "=", meaning "equal to") between the subject's security level ("SL(s)") and the object's security level ("SL(o)"). For objects with a security level range rather than a single security level, SLmax(o) ›= SLmin(o) is always true.

Access with respect to ring brackets is determined by the relationship (see page 51, "The Validation Level") between the subject's current validation level (VL) and the ring brackets of the object ("R1","R2","R3").

Each individual section is preceded by a discussion of the overall mandatory access rules that apply to all operations on the object, regardless of the specific access exercised.

Segment Access Modes

For segments, the mandatory access compatibility rule(1) implies that SL(o) for the segment is always equal to SL(o) for the containing directory.

---

(1) The "compatibility rule" is described in detail in the Multics Interpretation of the Bell and LaPadula Model (Bell, D. E. and LaPadula, L. J., Secure Computer Systems: Unified Exposition and Multics Interpretation, page 25, MTR-2997, Rev. 1, The MITRE Corp., Bedford, MA., March 1976).

read (r)	A process granted read access can execute instructions that cause data to be fetched (read) from the segment. A process may exercise read access if and only if: 1) the ACL term contains "r" 2) SL(s) >= SL(o) 3) VL <= R2
write (w)	A process granted write access can execute instructions that cause data to be input to a segment or existing data in the segment to be modified. A process may exercise write access if and only if: 1) the ACL term contains "w" 2) SL(s) = SL(o) 3) VL <= R1
execute (e)	A process granted execute access can transfer to the segment and cause portions of the segment to be interpreted and executed as instructions. A process may exercise execute access if and only if: 1) the ACL term contains "e" 2) SL(s) >= SL(o) 3) VL >= R1 4) VL <= R3
null (n)	The process cannot access the segment in any way.

Directory Access Modes

For directories, the mandatory access compatibility rule implies that SL(o) for a directory always dominates SL(o) for the containing directory.

status (s)

A process granted status access can obtain information about the attributes of the directory as well as attributes of segments, directories, and links contained in the directory.

A process may exercise status access if and only if:

1) the ACL term contains "s"
2) SL(s) >= SL(o)
3) VL <= R2

modify (m)

A process granted modify access to a directory can modify attributes of other objects (segments, directories, message segments, and links) contained in the directory, and delete objects from the directory. Access to the object itself is not considered in either case. The two exceptions are:

1) The bit count attribute is considered part of a segment's contents, and therefore requires "w" access to the segment whose bit count is to be changed, rather than "m" to the directory. The same is true of the damaged switch.
2) Objects managed in inner rings may not be modified or deleted except by software running in the inner ring.

A process may exercise modify access if and only if:

1) the ACL term contains "m"
2) SL(s) = SL(o)
3) VL <= R1

append (a)

A process granted append access can create new objects (directories, segments, message segments, links) in the directory.

A process may exercise append access if and only if:

1) the ACL term contains "a"
2) SL(s) = SL(o)
3) VL <= R1

null (n)

A process with null access to a directory can still reference objects contained in the directory to which it has non-null access, and can read the attributes of those objects, but cannot reference any other objects or attributes. If the directory is at a higher security level than the process (which means the process has null access to it), by definition, all the objects it contains are at a higher security level than the process, therefore the process cannot access any of them or sense their presence in any way. The exception is that a process attempting to delete a directory at a higher security level will succeed or fail depending on whether the directory is empty; however, all unsuccessful attempts are audited.

Message Segment Extended Access Modes

For message segments, the SL(o) is always the security level of the message, not the security level of the message segment. When a message segment is created, it is assigned a SLmax(o) equal to the maximum SL(s) permitted for the creating process. The message segment implementation ensures that SLmax(o) for the message segment always dominates SL(o) for any messages placed in the message segment. The mandatory access compatibility rule ensures, as for segments, that SL(s) dominates SL(o) for the containing directory in order to reference the message segment in any way. Therefore, since SL(o) for a message in a message segment is set equal to SL(s) for the subject adding it, SL(o) for all messages in a message segment must dominate SL(o) for the containing directory. No ring number checks are made on message segments.

In addition to the requirements listed below, for all message segment operations, a process may perform the operation if and only if:

A) SLmax(segment) >= SL(s)
B) SL(s) >= SL(dir)

add (a)

A process granted add access to a message segment can add a message.

A process may exercise add access if and only if:

1) the ACL term contains "a"
2) SL(o) >= SL(s)

delete (d)

A process granted delete access to a message segment can delete any accessible message.

A process may exercise delete access if and only if:

1) the ACL term contains "d"
2) SL(o) = SL(s)

read (r)

A process granted read access to a message segment can read any accessible message.

A process may exercise read access if and only if:

1) the ACL term contains "r"
2) SL(s) >= SL(o)

own (o)

A process granted own access to a message segment can read or delete its own accessible messages.

A process may exercise own access if and only if:

1) the ACL term contains "o"
2) Username(s) = Username(msg author)
3) SL(s) >= SL(o) (read msg)
4) SL(o) = SL(s) (delete msg)

status (s)

A process granted status access to a message segment can find out how many accessible messages are contained in the message segment (the count is the number of messages satisfying the rule below).

A process may exercise status access if and only if:

1) the ACL term contains "s"
2) SL(s) >= SL(o)

wakeup (w)
: A process granted wakeup access to a mailbox message segment can send a message to the mailbox and cause a receiving process to be notified. The message may be sent at a level greater than that of the process sending the message. This mode is valid only for mailbox message segments.

  A process may exercise wakeup access if and only if:
  1) the ACL term contains "w"
  2) SL(o) >= SL(s)

urgent (u)
: The urgent mode is equivalent to the wakeup mode (valid for mailbox message segments only) but causes the receiving process (if any) to receive an urgent wakeup.

  A process may exercise urgent access if and only if:
  1) the ACL term contains "u"
  2) SL(o) >= SL(s)

null (n)
: The process cannot access the message segment in any way.

RCP Access Modes

Each RCP resource is assigned a minimum (SLmin(o)) and maximum (SLmax(o)) security level, in addition to the current security level (SL(o)) it receives while in use. There is no mandatory access compatibility rule for RCP resources, because they do not constitute an object hierarchy. For all RCP operations, in addition to the requirements listed below, the operation may be performed if any only if:

A) SL(s) >= SLmin(o)
B) SLmax(o) >= SL(s)

read (r)
: A process granted read access to a resource can read the contents of the resource.

  A process may exercise read access if and only if:
  1) the ACL term contains "r"
  2) SL(s) >= SL(o)
  3) VL <= R2

write (w)
: A process granted write access to a resource can write the contents of the resource.

  A process may exercise write access if and only if:
  1) the ACL term contains "w"
  2) SL(s) = SL(o)
  3) VL <= R1

executive (e)
: A process granted executive access to a resource can change the attributes of the resource; in effect, granted the same rights to the resource as the resource owner.

  A process may exercise executive access if and only if:
  1) the ACL term contains "e"
  2) SL(s) = SL(o)
  3) VL <= R1

null (n)
: The process cannot access the resource in any way.

Logical Volume Access Modes

No mandatory access compatibility rule exists for logical volumes, because they do not constitute an object hierarchy. Logical volumes have an access range SLmax(o) >= SLmin(o), rather than a single security level. All segments on the logical volume are constrainted to have

A) SL(seg) >= SLmin(vol)
B) SL(seg) <= SLmax(vol)

read/write (rw)
: A process granted read/write access may request that the logical volume be mounted, and reference accessible segments on the logical volume. Both read and write must be present in the ACL term of the Access Control Segment (ACS) segment.

  A process may exercise read/write access if and only if:
  1) the ACL term contains "rw"
  2) SL(s) >= SLmin(o)
  3) SLmax(s) >= SL(o)

executive (e) — A process granted executive access to a logical volume can change the attributes of the logical volume as a volume administrator; in effect, granted the same rights to the resource as the logical volume owner.

A process may exercise executive access if and only if:
1) the ACL term contains "e"
2) SL(s) = SLmin(o)

null (n) — The process cannot access the logical volume in any way.

Communication Channel Access Modes

Each communication channel is assigned minimum (SLmin(o)) and maximum (SLmax(o)) security levels, in addition to the current security level (SL(o)) it receives while in use.

read/write (rw) — A process granted read/write access may attach the channel and transmit data across it. Both read and write must be present in the ACL term of the ACS segment.

A process may exercise read/write access if and only if:
1) the ACL term contains "rw"
2) SL(s) >= SLmin(o)
3) SLmax(o) >= SL(s)
4) SL(s) = SL(o) (assigned dynamically)
5) VL <= R1

null (n) — The process cannot access the channel in any way.

Process Access Modes

Processes are treated as objects for interprocess communication. Each process is assigned a security level, SL(s) for the sending process and SL(o) for the receiving process. No discretionary control for process objects is provided.

send wakeup (w)	A process with send wakeup access to another process may send a wakeup and a 72-bit data message to that process. A process may exercise send wakeup access if and only if: 1) SL(o) >= SL(s)
null (n)	The process cannot send a wakeup to the other process.

Discretionary Access Controls

This section describes the Multics Discretionary Access Control (DAC) mechanism. DAC allows users to grant or deny access to objects at their own discretion. In Multics, the access control list (ACL) is the mechanism that controls discretionary access. This type of access is useful to users who wish to share objects under their control with other users.

Access identifiers are used to identify users uniquely for the purpose of discretionary access and are fixed for the life of the user's process. Each identifier may be up to 32 characters in length and consists of three components: the name of the user on whose behalf the process was created, the user's project for the particular process creation, and an instance tag that identifies different classes of processes. A person can log into the system as a different user by using different projects and/or instance tags. For example, person Smith might be an interactive user on the Multics project, giving an access identifier of "Smith.Multics.a", while at the same time be running an absentee job from the SysLib project, resulting in an access identifier of "Smith.SysLib.m".

The ACL is an attribute associated with each object. It contains a list of process access identifiers and the associated access modes for each. An asterisk substituted for a component matches any value. A particular type of process belonging to a user can be specified by using all three components of the access identifier (e.g., rw Smith.Multics.a); all processes belonging to a user can be specified by an asterisk in the third component (e.g., rw Smith.Multics.*); an entire project can be specified by an asterisk in the first and last components (e.g., rw *.Multics.*) or all users can be specified by an asterisk in all components (*.*.*). All or any combination of components of the access identifier can be replaced by an asterisk, in order to grant or deny access to individual users or groups of users. In this way, granting and denying access to a group of users and

further restricting or extending access to particular users within the group is possible. Allowing asterisks to be substituted for components results in eight groups of access class identifiers. These identifiers are ordered from 1 to 8, with 1 being no asterisks and 8 containing all asterisks. In matching entries on any given ACL, the first match found specifies the discretionary access granted.

System-wide defaults are provided for directories ("sma" for the creator and *.SysDaemon.*); for data segments ("rw" for creator and *.SysDaemon.*); for executable segments ("re" for the creator and "rw" for *.SysDaemon.*); for mailboxes ("adrosw" for the creator, and "aow" for *.SysDaemon.* and *.*.*); and for message segments ("adros" for the creator and *.SysDaemon.*, and "aos" for *.*.*).

In addition to system-wide defaults, Multics provides a means for users to set up initial ACLs for directories and segments on a per-directory basis. When initial ACLs are provided by the user, a defined set of rules is followed to merge the different sets of ACLs. Initially, an "rw" term for *.SysDaemon.* is placed in the ACL for the newly created object. Next, the initial ACL for the object is added. Where access class identifiers are the same, the initial ACL overrides, allowing users to alter the access granted to the SysDaemon. Finally, the access mode for the creator specified in the creation of the object is merged in the same manner.

Users can create and maintain ACLs from the command level by using the list acl, set acl, and delete acl commands. Initial ACLs can be created and maintained in much the same manner with the list iacl seg, set iacl seg, and delete iacl seg commands for segment IACLs, and the list iacl dir, set iacl dir, and delete iacl dir commands for directories. These commands use a set of TCB subroutine interfaces to make the changes.

The access granted through the discretionary access control mechanism is known as the "raw access mode". The actual access mode that is enforced by the system is known as the "effective access mode" and is determined by applying mandatory (non-discretionary) access controls and intraprocess access controls (ring brackets) to the raw mode. For details, see page 27, "Objects Implemented by the TCB".

Mandatory Access Controls

Multics mandatory access control is handled by the Access Isolation Mechanism (AIM). Multics supports eight hierarchically ordered sensitivity levels and 18 categories that use set inclusion rules. System administration assigns character string names to the levels and categories that are used in a particular installation. AIM is always active on Multics systems, and the access checks are always being made. Its actions are nullified in those environments that are not interested in mandatory security by leaving the system's access ceiling at the default "system low". AIM checks begin having an effect when the system's access ceiling is set to some level and category set, and some users are registered at other than "system low".

The only subjects on Multics are processes. The three types of processes are interactive (those operating on behalf of interactive users), absentee (those created to run batch command files) and daemons (those used to perform system functions, including most of the privileged prooesses).

Each username (not user/project pair) has a default security level (authorization) associated with it. In addition, interactive users may request a specific level at login time by using the -auth control argument. Absentee processes are usually created at the level of the process that requested the absentee, although a higher level may be requested (a lower or disjoint level may not, as that would create a covert channel). Daemon processes are created at a level specified by the system administrator, or at their default level if none was specified.

Several sets of minimum and maximum authorizations determine the security level at which a process may be created. To create a process, the requested level (either the default, or an explicitly requested level) must fall between the following values:

the minimum and maximum authorizations specified in the user's entry in the Person Name Table (PNT),

the minimum and maximum authorizations specified in the System Administration Table (SAT) for the project being used by the user,

the minimum and maximum authorizations specified in the Project Definition Table (PDT) entry for the user,

and, for an interactive process, the minimum and maximum authorizations specified in the Channel Definition Table (CDT) for the channel being used to log in.

If the requested authorization does not fall within all of these ranges, the request for process creation is denied; otherwise, the process is created, and its current authorization is set to the requested authorization for the life of the process. The process maximum authorization is always set to the maximum value for the user in the PNT entry; the other maximum values are not included in this calculation.

The current and maximum process authorizations are stored in the Process Data Segment (PDS) for use by the supervisor. The current authorization is also stored in the Active Process Table Entry (APTE) for use by interprocess communication.

Each object is assigned a sensitivity label, called an AIM access class, when it is created (see page 27, "Objects Implemented by the TCB"). The sensitivity label consists of a level and a set of categories. A Multics object is either a segment accessible by user processes, or contained in a segment that is directly accessible only by the TCB, and accessible to users only through appropriate TCB interfaces. User-accessible segments are much like files on other systems. Segments accessible only by the TCB include directories, mailboxes, message segments, RCP registries, and other system tables. They are used to record and contain TCB-controlled resources: directories record the location and attributes of segments and other directories; message segments and mailboxes contain messages, each of which has an individual security label; RCP registries record the attributes of RCP-controlled resources; indeed, all TCB-controlled resources are recorded in some segment accessible only by the TCB. Most such segments are multi-class segments (see page 22, "Ring One TCB Functions"), because they contain information about objects at potentially different security levels.

In general, AIM enforces the following rules; for specific cases see page 27, "Objects Implemented by the TCB".

To read an object, the process authorization must dominate the security level of the object.

To write an object, the process authorization must be equal to the security level of the object (in general, writing includes reading in Multics).

To append to an object (only possible for message segments and IPC), the process authorization must be dominated by the security level of the object.

These access checks are done by trusted code including the ring zero TCB, ring one TCB, the administrative functions, the Answering Service (identification and authentication), and the privileged processes.

TCB Protection Mechanisms

The Multics TCB protects itself through a combination of the hardware protection mechanisms and specific software mechanisms. This section discusses the most important of these.

Use of Hardware Protection Mechanisms

The Multics hardware protection mechanisms described previously (see page 9, "Hardware Protection Mechanisms") are used by the ring zero supervisor and by the execution environment in all rings to protect the TCB from errors and malicious misuse. See page 46, "Software Protection Architecture", for a detailed discussion of their use by the TCB.

Use of CPU Protection Mechanisms

The set of data a process can reference at any instant is described by its descriptor segment (DSEG). This is a TCB-maintained segment identified to the processor by a hardware register, the Descriptor Base Register (DBR). The DSEG of a process consists of a series of Segment Descriptor Words (SDWs), each of which defines a segment accessible to the process, the rings in which the segment can be accessed, and the access modes

the process may exercise. The DSEG is the fundamental hardware reference monitor mechanism. All accesses to data are controlled by the information in the DSEG of a process.

At any time, an SDW may either be valid or invalid. If the SDW is valid, it is used directly by the hardware for accessing the data it describes. If it is invalid, an attempt to use it causes an implicit request to the TCB to make it valid so the process can continue. The TCB automatically invalidates the SDWs of segments whose access is changed, so that the next time a process attempts to reference such a segment, the TCB is forced (see page 48, "Access Revocation") to recalculate access.

The hardware rings provide a way to create protected subsystems. This mechanism is primarily used for parts of the TCB itself, which run in ring one, but can also be used for creating protected user applications. The hardware provides protection against ring violations, primarily by performing address validation automatically, but also by using the ring alarm register to properly maintain software validation levels and by prohibiting the transfer of control across ring boundaries except by using the CALL6 instruction. These mechanisms are described in detail in the Multics Processor Manual(1) and the Reference Guide.(2) The ring mechanism is a powerful means for extending the TCB.

Use of I/O Protection Mechanisms

The Multics Input/Output Multiplexer (IOM) and device controllers (MPCs) provide a versatile set of hardware protection mechanisms that allow user-written I/O control programs to be executed directly by the hardware without any possibility of protection compromise.

The IOM provides two protection modes (see page 14, "IOM Protection Mechanisms"). These two protected modes constrain the memory addresses that an I/O program may reference. The ring zero supervisor (in the I/O interfacer subsystem, IOI) sets up these constraints before initiating a user-supplied I/O program, by creating its own I/O program that first sets the constraints,

---

(1) DPS/Level 68 & DPS 8M Mutlics Processor Manual. Honeywell Information Systems, Inc., AL39-01B, February 1982.

(2) Multics Programmer's Reference Manual. Honeywell Information Systems, Inc., AG91-04, June 1985.

and then transfers to the first word of the user's I/O program. Since I/O initiation requires a privileged instruction, the TCB is always involved and can always set the appropriate constraints.

The TCB must also ensure that the constraints describe memory that belongs to the requesting process. For rel mode I/O, where I/O accesses are confined to a contiguous region of physical memory, the TCB forces the pages of the process' I/O buffer to be in an appropriate contiguous region of memory, and marks those pages as "wired" (wired pages cannot be evicted back to disk). For paged mode I/O, the TCB simply creates an I/O page table (the format is different from the page tables used by the CPU) describing the I/O buffer segment, and marks the pages as wired. Once this is done, the I/O program can be initiated, and the addressing constraints will ensure that it addresses only memory belonging to the process.

When the I/O program completes, it signals an interrupt to the CPU and saves I/O status in the process' I/O buffer. The TCB is invoked by the interrupt, and sends a message to the process, causing it to be scheduled for execution and informing it that the I/O is complete. The TCB also marks the I/O buffer as not in use, so the wired pages can be released. If the I/O program does not complete within a reasonable time (30 seconds), the TCB is invoked by a software time-out mechanism, and forcibly terminates the I/O, also informing the process. This prevents a process from accidentally (by referencing a broken device) or maliciously (by constructing a looping I/O program) monopolizing the I/O hardware and buffer memory.

Although a user I/O program cannot specify an IOM channel number or a different addressing restriction, it can specify a device number to select one of a set of devices (such as a set of tape drives) attached to a particular channel. The MPC, however, does not allow the device number to be changed once the I/O program has begun execution (see page 15, "MPC Protection Mechanisms"). To prevent the user from accessing a device for which he is not authorized, the ring zero supervisor prefixes the I/O program with an additional "no-op" command, whose sole purpose is to set the device number initially, so that the MPC will not allow it to be changed.

Because an I/O program that references device zero can later specify a different device number or modify the control store of the MPC itself, access to device zero is administratively restricted (using RCP) so that a non-privileged user is not permitted to use it. Only maintenance personnel are normally given discretionary access to device zero on any MPC. For tape

devices, normally all users are allowed discretionary access to all non-zero devices. Other MPC-attached devices are restricted to specific system processes (printers, for instance) or the TCB itself (disks).

Most TCB-controlled I/O (file system disk drives, the bootload console) uses absolute mode and does not use the IOM hardware protection mechanisms. Since the TCB constructs all the DCWs itself, no additional hardware protection is necessary.

Communication between the FNP and the central system, however, uses paged mode to restrict the FNP to referencing only the FNP communications buffer segment. This restriction is necessary because the FNP is attached by a "direct" channel, which allows the FNP to specify absolute memory addresses for I/O transfers, unlike most devices, which only perform transfers with the IOM generating all addresses. Thus, even in the event of a programming error within the FNP, it is restricted to accessing only its own buffer area, rather than arbitrary locations in system memory. This use of "least privilege" is desirable because of the asynchronous nature of the communications between the portion of the TCB software running in the FNP and the portion running in the CPU.

Software Protection Architecture

The Multics TCB is internally structured to minimize programming errors and to follow the principle of least privilege. This is done primarily by programming conventions for TCB modules: individual modules are expected to have well-defined functions and manipulate only those data structures for which they are responsible. This section discusses first the general architecture of software protection, and then some specific security-relevant protection functions used within the TCB. The TCB also, of course, depends on the hardware protection mechanisms, discussed above.

Although any module in the entire ring zero TCB potentially can modify data managed by another module, the use of segmentation to separate databases and the use of common include files to describe all TCB data structures minimizes this possibility. The same is true of the ring one TCB and the data it manages. The ring one TCB is further protected because it keeps much of its data in per-process storage that is not accessible at all by other processes running in ring one. Segmentation is quite effective as a protection against propagation of programming

errors; it is extremely rare that a shared database is damaged by any module other than the one normally responsible for maintaining it.

Within the TCB, the hardware access control meohanisms are used to ensure that program segments are never writeable and that data segments are never executable, protecting against errors(1) that might otherwise modify pure code or data. Additionally, these hardware mechanisms are used to allow only those ring zero programs that must execute privileged instructions to do so, providing an additional degree of separation of privilege. By and large, however, no attempt is made to allow access only when required, for instance for TCB data that is shared and frequently read, but rarely modified. TCB-controlled data within ring zero and ring one is also usually equally accessible to all processes, regardless of whether a process actually uses the TCB functions that reference the data.

The privileged processes are isolated from each other and from other processes by both their use of per-process storage and the ACLs on the shared or permanent data they maintain. This use of ACLs for isolation is perhaps the most fragile of the internal proteotion mechanisms, since it requires that a system administrator set all those ACLs properly, according to the specifications in the Trusted Facility Manual; if an ACL is changed for some reason (such as debugging), it usually must be explicitly reset.

The ring zero TCB is isolated from the ring one TCB and the privileged processes by the ring mechanism. Since, however, the ring one TCB and privileged processes must be able to request special services from the ring zero TCB, they have access to gates that request those services. Thus, if subverted, they could use those special services to subvert the ring zero TCB. However, this requires complex programming and could occur only in practice as the result of deliberate subversion, rather than from programming errors.

All access control decisions are made using the same set of primitives. The DAC primitives are used only in ring zero, the primitive takes the name of a file system object and returns the

---

(1) In practice, this happens only as a result of CPU hardware failures. Even during TCB software development, the structure of the system and the prevalence of PL/I coding makes this sort of accidental reference unlikely to result even from programming errors.

effective access specified by the appropriate ACL entry and ring brackets. The MAC primitives can be run in any ring. Given a pair of security levels, they return a result of "equal", "dominates", "is dominated by", or "disjoint". These primitives are the only places in the entire system where the internal formats of DAC and MAC information are interpreted to make access control decisions, ensuring that the correct decision rules are always used.

The results of an access control decision can be used directly by the hardware only for segments, and then only for actual data access. The results for all other objects or operations are further interpreted by software, and these decisions also are made in a single module for each object type. For each object that has DAC controls (segments, directories, message segments, RCP-controlled objects, and logical volumes) a single "access kernel" program is provided. These programs are told the name of the object and the requested operation, in terms appropriate to the particular object type, and return a "yes" or "no" decision. These access kernel programs call both the file system DAC primitives and the MAC primitives. For the remaining objects (messages, communication channels, interprocess communication), the yes/no access decision is made by calling the appropriate DAC and MAC primitives directly and interpreting the result appropriately. No separate access kernel programs are necessary, because the result from the primitives corresponds precisely to the decision required.

Decisions concerning the MAC-overriding AIM privileges (see page 51, "Mandatory Access Control Privileges") and ring brackets are not as well-defined, because the semantics of their protection is not as clearly specified. Although the access kernel programs can factor MAC-overriding AIM privileges into the decisions, this is not always requested (that is, these privileges are not a required parameter when making an access decision), and many programs make additional, separate decisions when factoring in privileges and ring brackets.

The remainder of this section provides a brief overview of the important mechanisms that provide the internal and user-visible protection features.

Access Revocation

When a segment's access is changed, the effect is seen immediately by all processes that have access to the segment. The effect is delayed for other types of objects, but only until

either the next attempt to perform an operation on the object, or until the object is released (detached, closed, etc.) by the process.

Access revocation, for segments, is primarily the responsibility of a mechanism called "setfaults", which removes an object whose access is changed from all address spaces in which it appears. If a process still has access to the object, it will be added back automatically (and invisibly) the next time it is used; otherwise, an invalid access error will be reported and the operation will fail.

Audit Message Logging

A general-purpose logging mechanism, called "syserr", is used by the entire TCB to record security-relevant events, hardware errors, internal inconsistencies, and other events of note. For robustness, part of this log is maintained outside the normal file system and is thus effectively immune to system failure. Once the system resumes normal operation, that part of the "syserr" log is automatically copied back out into segments. An arbitrarily long history can be maintained in this and other logs; it can even be stored on tape and selectively retrieved long afterwards.

The same basic software mechanism is used to implement all system logs; it incorporates robustness features to ensure that as few messages as possible are lost in the event of damage to a log segment. The major difference is that the syserr log is the only one maintained by the ring zero TCB outside the file system. Only ordinary segments are used for the Answering Service and admin logs.

Parameter Validation

To be secure, an operating system must ensure that it is not "spoofed" by invalid or changing values for the parameters passed to operating system calls. In Multics this is accomplished by a combination of hardware and software mechanisms. The hardware mechanism guarantees that when the supervisor references any data on behalf of a user request, it does so only with the effective privilege (ring) of the calling program. The software conventions govern argument reference and prohibit invalid multiple references.

The hardware mechanism used for parameter validation is the effective ring (see page 10, "Effective Ring Number in Addressing"). By requiring that all pointers used in any potentially multi-ring context be Indirect To Segment (ITS) pointers (the default PL/I "pointer" data type), which contain an effective ring field, the software ensures that any user-supplied address will be validated automatically each time it, or any derivative of it, is used. Configuration management assures that packed pointers, pointers not containing an effective ring field, may be used only in applications where addresses will never be used by multiple rings.

Various software mechanisms are also employed. The first to be encountered in any call to the supervisor is the check on number of arguments. This is performed immediately after the transfer to a gate segment changes the ring of execution, and ensures that a supervisor call will always receive the correct number of parameters.

The next is the restriction on parameter referencing. In general, user-supplied parameters must be referenced precisely once, since they reside in user storage, and the user can change their values between one reference and the next. This is usually accomplished by having all programs reached by cross-ring calls copy their input parameters into local storage before referencing them, since then they are proof against interference. The full rules for parameter reference are actually considerably more complex and described better in the referenced paper.(1)

These software conventions also apply to implicit parameters: data stored in a less privileged ring, on which more privileged code has dependencies (usually in order to perform a service more efficiently). This is a source of considerable complexity in the supervisor, since the rules are not nearly as obvious as for formal parameters. However, even if the software conventions are not followed, the hardware effective ring mechanism prevents protection errors.

The hardware effective ring mechanism is the major defense of the Multics TCB against internal programming errors. By enforcing the principle of least privilege in hardware, it eliminates a primary source of vulnerabilities in other operating systems.

---

(1) Bisbey, R., Popek, G. & Carstedt, J., Protection Errors in Operating Systems: Inconsistency of a Single Data Value Over Time, ISI/SR-75-4, ISI, December 1975.

The Validation Level

The validation level is the software equivalent of the hardware effective ring mechanism. It is a per-process value, maintained by the TCB, that records the level of privilege on behalf of which the TCB is operating.

For instance, in the same way that a process running in ring zero is not permitted to read a ring two segment because it was given the address by ring four, a process running in ring four cannot create a ring two segment because the ring zero TCB knows that the request came from a ring four process. It knows this because the validation level is set to four, and a process running in ring four is not permitted to set the validation level any lower than four.

Because the validation level is changed only by explicit action (a call to ring zero), it would be possible for a programming error (in ring one, for instance) to cause it to be set to one during a call to ring one, but not reset before the return. This is prevented by the hardware Ring Alarm Register (RALR), which signals an exception and invokes the TCB whenever an attempt is made to return to a ring less privileged than the current value of the RALR. The TCB sets the RALR every time it changes the validation level, and this ensures that it will always be reset to a valid value for the ring of execution, even if a more privileged program neglected to restore it.

Mandatory Access Control Privileges

Special users, normally the system security administrator, may enable privileges that nullify the effects of AIM in access computations. These privileges are:

ipc	Ignore AIM when sending wakeups to processes
dir	Ignore AIM when examining the contents of directories
seg	Ignore AIM when attempting access to segments
soos	Ignore the security-out-of-service switches set when the system detects AIM inconsistencies during crash recovery
ringl	Ignore AIM when accessing messages in message segments
rcp	Ignore AIM in RCP access computations
comm	Ignore AIM when assigning communications channels

These privileges are used by system processes and the system administrator for designated functions whose overall operation is trusted not to violate MAC constraints. For example, the backup and retriever daemons that create and use backup tapes must have seg and dir privileges to write data at all authorizations. To downgrade data (a necessary operation on any Mandatory Access Control system) the system security administrator must have the seg, dir, ringl, and/or rcp privileges. When a system failure results in a file system AIM inconsistency, the offending directory is marked "Security Out Of Service", and all attempts to access it by an unprivileged process fail. The soos privilege is needed by the system administrator in order to fix the damaged directory. The use of privileges is audited and can be invoked only through trusted software.

Unlike the interpretation of the basic DAC and MAC controls, which is centralized into basic primitives with a separate access control kernel for each object type, AIM privileges are not always interpreted by the access control kernels. This is true primarily because the semantics of the privileges are not as well-defined as the basic DAC and MAC controls. For those objects that have access kernel programs (see page 16, "Software Architecture"), the kernels interpret the appropriate privileges (seg, dir, soos, ringl, rcp). Those privileges, as well as comm and ipc, are also checked explicitly in other programs where they have some relevant meaning. This looser structure for privilege checking is not a problem, however, since privileges are used only by privileged processes.

Software Recovery From System Failure

Three major mechanisms are used to provide robust recovery from system failure: a protocol of updates to eliminate object reuse for segments (and thus all file system objects) when any type of failure occurs; a protocol for recovering file system databases after most crashes, allowing all modified data to be written back to disk; and finally, a set of salvagers to evaluate (after all crashes) and restore (after most crashes) consistency to various databases (directories and message segments, primarily).

The system automatically attempts to keep segments, directories, message segments, and messages within message segments consistent after system failure. Automatic mechanisms exist for detecting most types of damage to other TCB databases (RCP registries, the CDT, logical volume data), automatic salvaging is usually not performed, and administrative intervention is required to

reconstruct or retrieve this information. Internal consistency checks are present throughout the TCB to guard against undetected (or detected) damage causing a user to gain unauthorized access.

Object Reuse - Segments

Multics satisfies the object reuse requirement (for segments) by guaranteeing that whenever a page is referenced by an instruction, the result is either a page of data previously written into that page of the segment, or a page of zeros if that page had never been written to before. All segments are initially created empty, so that any references to their contents produces pages of zeros.

The object reuse requirement is satisfied during normal operation by keeping track of whether data has been written to a page or not. It is further guaranteed across all types of system and disk failures by keeping all information about a disk's contents on the disk itself, and maintaining that in such a fashion that a segment can never claim a page that does not contain information from that segment, and also by ensuring that a segment is always correctly identified. This is done as follows:(1)

> When on disk, a segment's contents are described completely by its VTOC entry. All the pages of any particular segment and its VTOC entry are kept on the same physical disk pack. The VTOC entry describes where to find those pages; the only information about a segment's contents that comes from the directory is the volume ID of the disk and the offset of the VTOC entry on that volume.
>
> The directory entry and VTOC entry for a segment must match (have the same unique identifier) for pages of the segment to be referenced at all. If they do not match, the reference is rejected at that point, and it is not even possible to reference pages of the segment.

---

(1) An equivalent mechanism is used to guarantee the proper allocation of the VTOC entries representing entire segments. It also uses a bit map and a stock as described here for records, and the logic described applies equally to both records and VTOCEs.

When a segment is in use and may have pages in memory, it is called "active". When in use, it is defined completely by its Active Segment Table Entry (ASTE), which identifies all its pages, in memory and on disk, and their addresses.

On every system disk is a bit map of free space that defines which records are available. When a segment grows, a record is removed from this map (by turning off its bit), and assigned to the segment. When a segment is deleted or shortened, the bits for all the records freed are turned back on. In normal circumstances, all bits in the bit map that are off (allocated) represent record addresses listed in VTOC entries on that volume, and no bits that are on (free) may represent record addresses in any VTOCE. During system operation, the stock mechanism (described below) is used to maintain an efficiently accessible logical image of the bit map in main memory.

The primary goal of the file system, then, is to ensure that no record marked as "free" in the bit map may appear in any VTOC entry. If this happens, the record could be assigned to another segment that wanted to grow, causing the data to appear in two places at once. This is called a "reused address".

The secondary goal is to ensure that no record marked as "allocated" in the bit map fails to appear in any VTOCE. If this happens, records can be lost irretrievably, since belonging to no segment, they will never be returned to the bit map.

The object reuse requirement is satisfied by the primary goal, but both goals must be achieved to provide satisfactory operation.

During normal operation, both goals are easily satisfied. Only in the case of system failure may the goals not be met, and by ensuring that at least the primary goal is always met, Multics satisfies the object reuse requirement even in the face of arbitrary system failure.

To reconstruct the bit map from all the VTOC entries after a system failure is comparatively easy. As long as addresses in the VTOC entries actually describe pages known to be on disk that belong to the segment, the bitmap can be rebuilt after a failure. This is accomplished by having a bit for each page of an active segment indicate whether that page has ever been written to its home on disk. If it has been written, then when the VTOC entry is written out to disk, it is safe to include that address in the VTOC entry. Otherwise, the VTOC entry will just indicate that the page does not exist. This ensures that whenever a VTOC entry

is written, the addresses it contains will identify pages on disk. The pages may be obsolete, if they were modified and the system failed, but at least they will belong to the segment. As long as the bitmap is rebuilt after a failure, its state at the time of the failure does not matter.

If the bit map is always rebuilt after a failure, both goals are easily met (since the VTOC is guaranteed to be consistent with the state of the records on the volume). However, rebuilding the bit map is expensive and to be avoided if at all possible. This is done by temporarily allowing the secondary goal to be missed and maintaining the bit map on disk so that it does not need to be rebuilt immediately after a failure. A "stock" of records is removed from the bit map on disk and updated to indicate that those records are all allocated. That stock is then kept by the TCB, parcelled out to segments as records are needed, and refilled as records are freed (but only after the VTOCEs from which they are freed have been written out). If it becomes empty, a new batch of records is allocated from the bit map on disk; if it becomes full, a batch of records is returned to the bit map as free. Thus, the bit map on disk will at any time indicate that some records are allocated when they are actually not in use. If a system failure occurs, the in-memory stock will be lost, and those records will be temporarily unavailable. However, as long as the stock is relatively small, this is a benign condition, and it is safe to rebuild the bit map after the system is running again, during normal operation.

Although these inconsistencies cannot arise during normal operation, it is possible for them to be created by a minor hardware problem or media failure (such as a disk error when writing a VTOC entry). If an allocation inconsistency is detected during normal operation, that disk is marked as damaged, and no new segments are created on it. It can then be repaired by rebuilding the bit map during system operation.

Object Reuse - Non-Segment Objects

The file system protects against object reuse for segments, and thus for all file system based objects, by using a mechanism for allocating records that guarantees a page will never be invalidly reused. Since all TCB-protected objects are implemented using segments (except for RCP-controlled devices and volumes, and objects without permanent memory), and since the managers for the objects themselves have a robust internal allocation mechanism, this automatically extends the basic object reuse protection to all objects.

Safe Shutdown and the File System

In Multics, where the virtual memory is the only access method for permanent files, keeping those files consistent is very important, even in the event of system failure. Because a file may at any time be partly represented in memory and partly on disk, the ability to write modified pages of memory back to disk, even after a system failure is important. Although this cannot be done after all failures, it is possible in most cases, through a procedure called "emergency shutdown" (ESD). The job of ESD is to write out all modified pages and VTOC entries. No attempt is made to keep higher-level objects consistent, but merely ensuring that all permanent data are completely up to date, even if inconsistent due to some interrupted operation, is valuable.

Because the system has failed at an arbitrary point in its operation, ESD cannot make assumptions about the state of TCB databases when writing out modified data. Instead, it must force them to be consistent and then write out the data. This is possible because the TCB keeps redundant information and always updates it in a defined order. For instance, when writing out a page, it marks the page as "being written" and starts the I/O; only when the I/O is completed does it mark the page as "written". Thus, ESD need not worry about whether the I/O completed or not; it just discards all pending I/O operations and restarts them for pages that are "being written". However, rather than using special mechanisms for all this, the only special mechanism is the one that puts the file system databases back into a consistent state, based on the redundant information. From that point on, ESD uses the same mechanisms as are used in normal operation to write things out, knowing that they will be written consistently.

These rules allow ESD to write out all modified pages, update all modified VTOC entries, and keep quota information consistent. If ESD fails, some or all of these things may not be done and will need repair when the system comes back up. However, as described above, even if ESD fails, object reuse is not a problem.

File System Salvaging and Consistency

The file system automatically detects and corrects inconsistencies in directories, ensuring wherever possible that MAC constraints are not violated even if an inconsistency has arisen due to system failure. The directory salvager, as well as

some other file system repair programs that correct more esoteric inconsistencies, can be run at any time during normal system operation as directed in the Trusted Facility Manual.(1)

Although no automatic recovery mechanism is provided for interrupted directory operations, they are automatically detected, and the directory is rebuilt. Directories contain a large amount of redundant information, and the directory salvager must examine a directory for inconsistencies and rebuild it consistently, possibly removing anything inconsistent.

When a directory modification is begun, the directory is marked as modified, and when finished, that indicator is reset. If any attempt is made to use a directory and that indicator is found on, the directory is automatically salvaged. The same occurs whenever an inconsistency is encountered during a directory operation, the directory is salvaged, and the operation automatically retried.

Particular care is taken with the sensitivity label (AIM) marking for segments in a directory. If the salvager finds that the AIM label of a segment does not match the AIM label in the segment's VTOC entry, the segment is marked security-out-of-service, a message is logged for the system administrator, and the security administrator must resolve the inconsistency manually before the segment can be accessed again. The same action is taken for inconsistencies that could create covert channels. ACL inconsistencies log a message, but do not cause the segment to be set out-of-service.

Message segments have a similar salvager, as do the RCP registries (in which all RCP objects are recorded) and the CDT (in which communication channels are recorded), although not as many types of inconsistencies can be detected because there is not as much redundant information. These objects depend more on basic file system recovery (ESD) to keep them consistent.

Covert Channel Management

Covert channels in Multics are controlled in one of four ways: (1) total closure by explicit programming, (2) bandwidth reduction by explicit programming, (3) auditing, and (4)

---

(1) Multics System Administration Procedures, Part VI and Appendix B, Honeywell Information Systems, Inc., AK50-03A, July 1985.

bandwidth reduction by the general "covert channel bandwidth limiter". These methods are applied to all covert storage channels and to many covert timing channels.

The first method is used to explicitly forbid operations that would create covert channels, such as being able to sense the existence of an object to which MAC forbids all access. Many controls of this sort are in the file system and other TCB subsystems.

The second method is used when the operation cannot be explicitly forbidden, but where the amount of information transferred can be reduced to a negligible amount. This is not used significantly in Multics, partly because of the difficulty of making reliable bandwidth estimations, and also because of the simpler and more general solution provided by the fourth method.

The third method is used primarily for events that are already quite slow and cannot be easily forbidden. Examples include exhaustion of space on entire disk packs and attaching and detaching shareable tape drives. These operations are all relatively slow to begin with, and the huge volume of audit messages generated by any attempt to transfer a significant amount of information will alert the system administrator as soon as the audit logs are examined.

The fourth method provides an overall limit on information transfer without explicitly forbidding any specific operation. For instance, the size of a mailbox can be manipulated by putting messages in and taking them out; although the messages themselves will not be accessible to the receiving process, the number of records used by the mailbox will be. The covert channel "event" in this case is the creation or destruction of a page in a multi-class segment.

Forbidding such events is not reasonable; mailboxes grow and shrink under normal conditions. Installing a specific time delay after each such event is equally unreasonable, since in most cases the event is not an attempt to use a covert channel, and such a delay would just waste time. Also, since a large class of potential covert channel events exists, installing an appropriate delay after each one would be difficult.

Instead, Multics provides a global covert channel bandwidth limiter (for each process) that works as follows. Each potential covert channel event invokes the limiter and informs it that an event has occurred. Every time 100 events occur in a process, the limiter checks the length of time they took, and if it is less than 1 second (more than 100 events per second), the process

is delayed until the bandwidth falls under 10 bits per second (these threshold values are adjustable), and an audit message is produced.

Because all events are considered together, this mechanism imposes a limit on all covert channels at once. A process cannot use any combination of channels to transfer more than 100 bits per second, yet as long as it stays under that threshold, no extra delays or special mechanisms are required to control those events (beyond incrementing the counter and checking the bandwidth). This allows the system to run just as efficiently as if no controls were being applied to these channels, unless, of course, a user attempts to exploit them.

Some timing channels are closed by this mechanism, and others are closed explicitly. Although no specific action is taken to eliminate the timing channels in the scheduler and page control, in general these are extremely noisy and quite slow. Unlike a more conventional system in which a program has some knowledge of physical devices and physical memory (even if virtualized), in Multics, a program has so little knowledge of its physical environment that the classic timing channels are very difficult to exploit and very noisy.

This page intentionally left blank.

EVALUATION AS A B2 SYSTEM

Discretionary Access Control

Requirement

> The TCB shall define and control access between named users and named objects (e.g., files and programs) in the ADP system. The enforcement mechanism (e.g., self/group/public controls, access control lists) shall allow users to specify and control sharing of those objects by named individuals, or defined groups of individuals, or both. The discretionary access control mechanism shall, either by explicit user action or by default, provide that objects are protected from unauthorized access. These access controls shall be capable of including or excluding access to the granularity of a single user. Access permission to an object by users not already possessing access permission shall only be assigned by authorized users.

Applicable Features

The Multics TCB defines and controls access between users and objects with access control lists that allow identification of users individually or by groups.

System-wide defaults are provided and grant access only to the object creator and the system daemon processes. However, an authorized subject has the ability to create default access control lists on a per-directory basis for both directories and segments by using the set iacl seg and set iacl dir commands. Multics provides a set of rules for merging system defaults with user-provided defaults.

Multics provides a group mechanism in the form of projects. Access can be granted or denied to an entire project. In addition, the object creator has the ability to further restrict or extend access to individuals within a project.

In order to grant access to an object to other users (change the object's access control list) a user must have modify access to the containing directory. A subject grants access rights to

objects by using the set acl command, while access rights to objects can be removed by using the set acl or delete acl commands. Both commands are invoked at the user interface level.

Conclusion

Multics MR11.0 satisfies the B2 Discretionary Access Control requirement.

Additional Requirement (B3)

The following changes are made in this requirement at the B3 level:

> CHANGE: The enforcement mechanism (e.g., access control lists) shall allow users to specify and control sharing of those objects. These access controls shall be capable of specifying, for each named object, a list of named individuals and a list of groups of named individuals with their respective modes of access to that object.
>
> ADD: Furthermore, for each such named object, it shall be possible to specify a list of named individuals and a list of groups of named individuals for which no access to the object is to be given.

Conclusion

Multics MR11.0 satisfies(1) the additional provisions of the B3 Discretionary Access Control requirement. Multics ACLs can include an arbitrary number both of users and groups (projects), each of which can either be granted or denied access.

---

(1) Although Multics MR11.0 satisfies this requirement at the B3 level, it does not satisfy any of the assurance requirements above its rated level.

Object Reuse

Requirement

> When a storage object is initially assigned, allocated, or reallocated to a subject from the TCB's pool of unused storage objects, the TCB shall assure that the object contains no data for which the subject is not authorized.

Applicable Features

When considering object reuse, Multics objects fall into four categories: user-accessible segments, information recorded in segments accessible only to the TCB, RCP-controlled physical disk and tape volumes, and memoryless objects.

Object reuse for user-accessable segments is satisfied under both normal operating conditions and most types of hardware or system failure by the mechanism described above (see page 53, "Object Reuse"). This ensures that all data read from a segment was previously written to that same segment. Even though the data in a deleted segment still remains on disk, the record allocation mechanism prevents it from ever being accessed until it has been overwritten with other data. Since all access to segments is made through the virtual memory, and users never have direct physical access to disk, this satisfies the object reuse requirement.

The TCB relies on this mechanism to make the same guarantee for information in the segments that are used to implement other TCB-controlled objects. Additionally, the TCB explicity clears or invalidates entries in these segments as part of the operation of deleting the corresponding object. Even if some pages in a TCB segment are destroyed by a system crash and others are not, the internal consistency checks within the TCB ensure that a deleted object will never be reused.

For RCP-controlled volumes, to meet the object reuse requirement, the Resource Management (RM) option of RCP must be enabled and the "manual clear" option enabled for all RCP volumes. When the RM option is enabled, removable volumes (tapes and disks, which are managed by RCP) are also protected from object reuse, because RCP requires manual intervention before a volume released by a user can be assigned to and used by another user. The system administrator is required to degauss the volume and report that operation to the system before the volume can be reused.

When an authorized user acquires an RCP-controlled volume from the free pool, the owner field in the RCP registry is set to the user's ID, and the access class is set to the user's current authorization. When a user no longer needs the volume, the volume can be released, an operation permitted only for the volume owner. When the owner issues the command to release the volume, the system responds with a warning that it may be degaussed, and the user must respond that this action is acceptable. The owner field in the RCP registry is set to "free", and a flag is set to indicate that the volume is "awaiting clear" state. The system recognizes the flag, and prevents other user from acquiring the volume. When the flag is turned off by the system administrator after degaussing, the volume can again be acquired by a new authorized user.

Memoryless objects (RCP-controlled devices, communication channels, and interprocess communication) have no means of storing information, and therefore the object reuse requirement is not applicable. For RCP-controlled devices, any information transferred is stored on an RCP-controlled volume, which is handled as described above. Communication channels have no inherent storage capacity; although they may be attached to some device that does, that attachment is outside the TCB boundary. Interprocess communication is a one-way, write-only, path, and cannot return information.

Conclusion

Multics MR11.0 satisfies the B2 Object Reuse requirement.

Labels

Requirement

> Sensitivity labels associated with each ADP system resource (e.g., subject, storage object) that is directly or indirectly accessible by subjects external to the TCB shall be maintained by the TCB. These labels shall be used as the basis for mandatory access control decisions. In order to import non-labeled data, the TCB shall request and receive from an authorized user the security level of the data, and all such actions shall be auditable by the TCB.

Applicable Features

Segments, directories, and message segments all have labels which are stored in the containing directory and VTOCEs. Individual messages are stored in message segments, and the label on each individual message is stored in the message header.

The only imported media are RCP volumes. All RCP-controlled volumes (e.g., tapes and disks) must be registered before they are recognized by the system. Part of this registration process involves setting a potential access class range for the volume, which must be within the range of the system defined defaults. Once the volume is registered, it is known to the system. Prior to use, the volume must be acquired. When a volume is first acquired, the actual access class range is recorded in the registry for this volume, provided that the access class range is within the potential access class range. This actual access class range is set to include only the current authorization of the process and cannot be changed except by a privileged user. This action is audited. Everything on a given physical volume is treated in the same manner, thereby making it a single-level object. Resource Management (RM), an option of RCP, must be enabled for mandatory access control to be enforced on RCP-controlled volumes and devices.

A process' label, the process authorization, is determined at login time (see page 41, "Mandatory Access Controls") and is not allowed to change for the life of the process. If one wants to change the label of a process, the process must be destroyed and a new one created with the desired label. This label is kept in a number of system tables.

Conclusion

Multics MR11.0 satisfies the B2 Labels requirement.

Label Integrity

Requirement

> Sensitivity labels shall accurately represent security levels of the specific subjects or objects with which they are associated. When exported by the TCB, sensitivity labels shall accurately and unambiguously represent the internal labels and shall be associated with the information being exported.

Applicable Features

Directory, segment, and message segment labels are stored in directories and Volume Table Of Contents Entries (VTOCEs). Since these can be manipulated only by the TCB, these labels are guaranteed to be protected from unauthorized modification. The labels on individual messages are stored in the message header, which is in turn stored with the message in a message segment. Since message segments can be manipulated only by the TCB, these labels are guaranteed to be protected from unauthorized modification. The labels on RCP-controlled volumes are stored in segments known as RCP registries. These registries are also protected from unauthorized modification. Trusted software handles all exportation of information and correctly associates the labels with the information being exported.

Conclusion

Multics MR11.0 satisfies the B2 Label Integrity requirement.

Exportation of Labeled Information

Requirement

> The TCB shall designate each communication channel and I/O device as either single-level or multilevel. Any change in this designation shall be done manually and shall be auditable by the TCB. The TCB shall maintain and be able to audit any change in the current security level associated with a single-level communication channel or I/O device.

Applicable Features

Each communication channel has an entry in a per-system table, the Channel Definition Table (CDT), which describes its various attributes. All changes to the CDT are made by creating a new CDT and submitting it to the TCB for installation. Part of this installation process includes generating the necessary audit messages documenting how the CDT has been changed. Part of the entry in the CDT for each communication channel is an allowable range of security levels that can be processed. However, when a channel is in use (i.e., assigned to a process), it is allowed to operate only in a single-level mode, and that level must be

within the specified range. Communication devices can be connected to Multics only over such a channel, therefore the devices are implicitly labeled.

A similar mechanism, the RCP registries, is provided for RCP controlled devices. Changes to the registries are audited by the RCP access kernel that is part of the TCB and controls all access to the registries. As with communication channels, RCP devices are allowed to operate only in a single-level mode when in use.

On the system disks, the allowable range of security levels is stored in the disk volume header. If the maximum level equals the minimum level, the device is still treated as a multilevel device in that the labels on the information are still trusted; it will simply have only one level of information stored on it.

Backup tapes are handled entirely by trusted software and are therefore able to store any information on the system. Since the labels are written and read only by trusted software, they can reside on the tape and still be trusted, thereby making backup tapes multilevel objects.

Printers are multilevel devices whose security ranges are defined in the I/O Daemon Table (IODT). This table may be changed only by a system administrator. Changes to this table are audited only as normal file accesses are audited. Therefore, the auditor must pay special attention to audit messages regarding access to this segment.

Conclusion

Multics MR11.0 satisfies the B2 Exportation of Labeled Information requirement.

Exportation to Multilevel Devices

Requirement

> When the TCB exports an object to a multilevel I/O device, the sensitivity label associated with that object shall also be exported and shall reside on the same physical medium as the exported information and shall be in the same form (i.e., machine-readable or human-readable form). When the TCB exports or imports an object over a multilevel communication channel, the

> protocol used on that channel shall provide for the unambiguous pairing between the sensitivity labels and the associated information that is sent or received.

Applicable Features

The only multilevel devices and objects on Multics are system disk packs (i.e., disks used directly by the TCB), backup tapes, and printers. Segments, directories, and message segments are kept only on system disk packs, and, as described on page 65, "Label Integrity", their labels are kept in directories and VTOCEs. The VTOCEs, and therefore the labels, are kept on the same medium. Backup tapes are handled only by trusted software, and the labels are kept on the tape with the data.

Printers are controlled entirely by trusted software that prints the proper label on each object. However, the printer software cannot prevent users from printing their own version of the system banner pages in addition to the system version. User-produced banner pages can be a source of confusion and error on the operator's part when the objects are burst and distributed. To alleviate this problem, the printer software prints on an operator's console a message that contains the serial number of each object. The operator matches messages from this terminal with the objects being burst before distributing any object. Additionally, a terminal known as a "forms control" terminal can be established for the sole purpose of providing information to help the operator correctly separate objects. For every object sent to the printer, a message is printed on this special terminal. This message contains the serial number of the object, the classification, and any other information that the site feels would be useful.

No multilevel channels exist on Multics.

Conclusion

Multics MR11.0 satisfies the B2 Exportation to Multilevel Devices requirement.

Exportation to Single-Level Devices

Requirement

> Single-level I/O devices and single-level communication channels are not required to maintain the sensitivity labels of the information they process. However, the TCB shall include a mechanism by which the TCB and an authorized user can reliably communicate to designate the single security level of information imported or exported via single-level communication channels or I/O devices.

Applicable Features

The single-level devices on Multics are communication channels (typically attached to terminals) and RCP-controlled devices. Via the trusted path, the user requests the security level for the process at process creation time. Provided that the level is within appropriate bounds (see page 41, "Mandatory Access Controls"), a process is created at that level, and the security level of the process's terminal is set to that level. Any other communication channels assigned to the process are set to the same security level. For RCPRM devices, the device is set to the level of the process when the device is attached, provided that the process is within the potential access class range for the device.

Conclusion

Multics MR11.0 satisfies the B2 Exportation to Single-Level Devices requirement.

Labeling Human-Readable Output

Requirement

> The ADP system administrator shall be able to specify the printable label names associated with exported sensitivity labels. The TCB shall mark the beginning and end of all human-readable, paged, hardcopy output (e.g., line printer output) with human-readable

> sensitivity labels that properly(1) represent the sensitivity of the output. The TCB shall, by default, mark the top and bottom of each page of human-readable, paged, hardcopy output (e.g., line printer output) with human-readable sensitivity labels that properly represent the overall sensitivity of the output or that properly represent the sensitivity of the information on the page. The TCB shall, by default and in an appropriate manner, mark other forms of human-readable output (e.g., maps, graphics) with human-readable sensitivity labels that properly represent the sensitivity of the output. Any override of these marking defaults shall be auditable by the TCB.

Applicable Features

The printable label names that are associated with exported labels are those that the security administrator designates for the levels and categories. The printer software always prints the security label on the header and trailer pages of all hardcopy output and additionally on the special "forms control" terminal (if configured). By default, each individual page of hardcopy output is marked top and bottom with the security label of the subject requesting the hardcopy output. Users are allowed the option to override individual page marking, but the override is audited.

Conclusion

Multics MR11.0 satisfies the B2 Labeling Human-Readable Output requirement.

---

(1) The hierarchical classification component in human-readable sensitivity labels shall be equal to the greatest hierarchical classification of any of the information in the output that the labels refer to; the non-hierarchical category component shall include all of the non-hierarchical categories of the information in the output the labels refer to, but no other non-hierarchical categories.

Subject Sensitivity Labels

Requirement

> The TCB shall immediately notify a terminal user of each change in the security level associated with that user during an interactive session. A terminal user shall be able to query the TCB as desired for a display of the subject's complete sensitivity label.

Applicable Features

Subject (process) labels cannot change during the life of a process. When a process is created, the TCB displays a message on the terminal indicating the security level of the process; this message cannot be overridden. The print process auth command is available during a session to query the TCB for the process label; a direct TCB subroutine interface is also available. Although a user may initiate multiple processes during a single terminal session, if (as recommended in the Trusted Facility Manual)(1) the strict trusted path system parameter is set, all those processes will have the same level. In order to create a process at a different level the user must login again using the trusted path.

Conclusion

Multics MR11.0 satisfies the B2 Subject Sensitivity Labels requirement.

---

(1) Multics System Administration Procedures, Part VI and Appendix B, Honeywell Information Systems, Inc., AK50-03A, July 1985.

Device Labels

Requirement

> The TCB shall support the assignment of minimum and maximum security levels to all attached physical devices. These security levels shall be used by the TCB to enforce constraints imposed by the physical environments in which the devices are located.

Applicable Features

As described on page 67, "Exportation to Multilevel Devices", the TCB maintains tables (CDT, IODT, RCP registries, and volume labels) containing the maximum and minimum levels of information that can be processed on each device or channel resource. These levels are compared with the subject labels of processes attempting to reference the resource, and the reference is disallowed if the process level is outside the range defined for the resource. Additionally, to ensure validity, the maximum level in the range must dominate the minimum level.

Conclusion

Multics MR11.0 satisfies the B2 Device Labels requirement.

Mandatory Access Control

Requirement

> The TCB shall enforce a mandatory access control policy over all resources (i.e., subjects, storage objects, and I/O devices) that are directly or indirectly accessible by subjects external to the TCB. These subjects and objects shall be assigned sensitivity labels that are a combination of hierarchical classification levels and non-hierarchical categories, and the labels shall be used as the basis for mandatory access control decisions. The TCB shall be able to support two or more such security levels. The following requirements shall hold for all accesses between all subjects external to the TCB and all objects directly or indirectly accessible by these

> subjects: A subject can read an object only if the hierarchical classification in the subject's security level is greater than or equal to the hierarchical classification in the object's security level and the non-hierarchical categories in the subject's security level include all the non-hierarchical categories in the object's security level. A subject can write an object only if the hierarchical classification in the subject's security level is less than or equal to the hierarchical classification in the object's security level and all the non-hierarchical categories in the subject's security level are included in the non-hierarchical categories in the object's security level.

Applicable Features

The Multics TCB enforces a mandatory access control policy over all resources that are directly or indirectly accessible by subjects external to the TCB. Subjects and objects are labeled as described previously (see page 27, "Objects Implemented by the TCB").

Multics supports 8 hierarchically ordered sensitivity levels and 18 categories which use set inclusion rules. The rules for accessing objects are:

1) A process may read an object only if the process authorization dominates the object access class.

2) A process may write to an object only if the process authorization is equal to the object access class, since write access to objects in Multics in general implies read permission as well.

3) A process may append or send messages to an object if the object access class dominates the process authorization; this rule applies only to message segments and IPC channels.

The exact application of the above rules between subjects and all object types is described in The Multics MR11.0 interpretation of the Bell and LaPadula Model.(1)

---

(1) Bell, D. E. and LaPadula, L. J., Secure Computer Systems: Unified Exposition and Multics Interpretation. MTR-2997, The MITRE Corp., Bedford, MA., July 1976.

Conclusion

Multics MR11.0 satisfies the B2 Mandatory Access Control requirement.

Identification and Authentication

Requirement

> The TCB shall require users to identify themselves to it before beginning to perform any other actions that the TCB is expected to mediate. Furthermore, the TCB shall maintain authentication data that includes information for verifying the identity of individual users (e.g., passwords) as well as information for determining the clearance and authorizations of individual users. This data shall be used by the TCB to authenticate the user's identity and to determine the security level and authorizations of subjects that may be created to act on behalf of the individual user. The TCB shall protect authentication data so that it cannot be accessed by any unauthorized user. The TCB shall be able to enforce individual accountability by providing the capability to uniquely identify each individual ADP system user. The TCB shall also provide the capability of associating this identity with all auditable actions taken by that individual.

Applicable Features

The Answering Service subsystem of the TCB is responsible for the identification and authentication of users before they are allowed to use the system. Two classes of requests (see page 76, "Trusted Path") are available from the Answering Service before a user logs in: authenticating and informational. The authenticating requests (described below) are used to create a process or connect to an existing process, and require that a user identity and password be presented. Informational requests may be used prior to an authenticating request, to set or determine terminal characteristics and obtain information about logging in. No authentication is required for these because they do not perform any subject-specific actions, although they do

represent part of the trusted path because they can display information about terminal security level, and other trusted information.

A standard user process is created by the login request. The user may log in at a default authorization level or use a special login argument (-auth) to request a particular authorization which, along with other parameters, will set a process's authorization (see page 41, "Mandatory Access Controls"). The login request can also be used to connect to an existing process that has been "suspended"; that is, which has been disconnected from the terminal at which it was previously connected to.

Passwords are required for login. Passwords may be up to eight characters long. The Answering Service also handles password changing, and can generate a random, pronounceable password upon request or if required by the system administrator. A randomly generated password is based on the hardware clock, which provides an essentially unlimited domain for random passwords, as it increments once per microsecond. The length of generated passwords may be set by the system administrator to between five and eight characters; the default is six.

A special type of process, called an "anonymous user", may be created through the enter and enterp requests. Any number of anonymous users are allowed to log in under the same identifier and need not necessarily supply a password. Use of the anonymous user option can be restricted by the system administrator. Two other commands, the dial and slave requests, allow users to connect a terminal to an existing process that may already have a terminal connected to it. To provide individual accountability, the system administrator can require users of these requests to identify and authenticate themselves. To meet the B2 Identification and Authentication requirement, the system administrator must not allow anonymous users and must require authentication of dial and slave requests.

Four system tables are referenced before a user can gain access to the system: the Person Name Table (PNT), containing the name, authorization range, and an encoded form of the passwords for every user; the System Administration Table (SAT), containing the authorization information for every project; a Project Definition Table (PDT), containing the individual names and per-project authorizations; and the Channel Definition Table (CDT), containing the authorization restrictions for the channel. Only the TCB can modify these tables, and modifications are audited. Only system administrators can read these tables, and they may be modified only through privileged TCB interfaces.

A unique user identifier exists for every user on the system and is associated with every process executing on the user's behalf. In addition, the user's current project and the process class (i.e., interactive, daemon, or absentee) are associated with the process. The TCB stores these attributes in the process's Active Process Table Entry (APTE) and in the Process Data Segment (PDS). When a process performs an auditable event, the TCB records the corresponding user identifier and project in the audit record.

Several administrative control features are available for passwords. The system administrator may force all users to change their passwords within a specifiable interval; use only system-generated random passwords; select only passwords exceeding a certain minimum length if user-chosen passwords are allowed; specify the length of system-generated random passwords; and force users to request system administrator revalidation if their passwords have not been used within a specifiable interval. These mechanisms are all provided by system parameters identified in the Trusted Facility Manual.(1)

Additionally, the system administrator may lock individual passwords, preventing their use; force individual users to change their passwords; force individual users to use system-generated random passwords; and cause an audible alarm at the system console when an individual's password is used.

Conclusion

Multics MR11.0 satisfies the B2 Identification and Authentication requirement.

Trusted Path

Requirement

> The TCB shall support a trusted communication path between itself and users for initial login and authentication. Communications via this path shall be initiated exclusively by a user.

---

(1) Multics System Administration Procedures, Part VI and Appendix B, Honeywell Information Systems, Inc., AK50-03A, July 1985.

Applicable Features

The Data Terminal Ready (DTR) signal can be used to provide a trusted path to the TCB. The DTR signal is generated (and stays on) when a terminal is turned on or a modem connection is made on a channel. The Answering Service uses that signal to start the login procedure for a new user. Once a trusted path is established, the user may execute informational requests(1) and then an authenticating request (page 74, "Identification and Authentication").

If the DTR signal is dropped by turning off a terminal or hanging up a phone, the terminal connection is broken and any process connected to that terminal is disconnected or terminated. A disconnected process can only be resumed if the owner logs in again and chooses to reconnect.

The system administrator can set a parameter to disable the use of the logout -hold and new_proc -auth commands to eliminate the possibility of a "spoofing" program and to force users to use the trusted path to change user identity or process sensitivity level. For the system to meet the B2 Trusted Path requirement, this parameter must be set.

Conclusion

Multics MR11.0 satisfies the B2 Trusted Path requirement.

Audit

Requirement

> The TCB shall be able to create, maintain, and protect from modification or unauthorized access or destruction an audit trail of accesses to the objects it protects. The audit data shall be protected by the TCB so that read access to it is limited to those who are authorized for audit data. The TCB shall be able to record the following types of events: use of identification and authentication mechanisms, introduction of objects into a user's address space

---

(1) Multics Commands and Active Functions. Part IV, Honeywell Information Systems, Inc., AG92-06, February 1985.

> (e.g., file open, program initiation), deletion of objects, and actions taken by computer operators and system administrators and/or system security officers. The TCB shall also be able to audit any override of human-readable output markings. For each recorded event, the audit record shall identify: date and time of the event, user, type of event, and success or failure of the event. For identification/authentication events the origin of request (e.g., terminal ID) shall be included in the audit record. For events that introduce an object into a user's address space and for object deletion events the audit record shall include the name of the object and the object's security level. The ADP system administrator shall be able to selectively audit the actions of any one or more users based on individual identity and/or object security level. The TCB shall be able to audit the identified events that may be used in the exploitation of covert storage channels.

Applicable Features

Multics records audit events in one of three logs: the syserr log, the Answering Service log, or the admin log. All audit messages are generated by the TCB code that attempts to perform an action on behalf of a process. The audit logs are protected from unauthorized access, modification, and destruction by DAC and rings.

The syserr log contains the audit records for object manipulations (e.g., introduction into address space, deletion, ACL change). In addition, it contains the audit records describing modifications to system databases and the audit records generated by the use of identified covert channels. Messages in this log contain the date and time of the event, the name of the system module that generated the message, the name and authorization of the user on whose behalf it was acting, the name and security label of the object, the type of operation requested, the success or failure of the action, and any special privilege that was enabled at the time of the event.

The Answering Service log contains the audit records for all events that relate to process management (e.g., login, process destruction) and events relating to use of communication channels. The messages in this log contain the date and time of the event, the type of event attempted, the success or failure of

the event, the communication channel identifier associated with the action, and the name of the user associated with the process attempting the action.

The admin log records all terminal I/O performed on the operator's console, including actions taken by the security administrator using the console in "admin mode". Messages in this log contain the date and time of the action, the name of the user or ohannel identifier requesting the action, the request itself, and all output produced as a result of the request.

Multics provides both selective recording of auditable events and post-audit reduction tools. A set of flags indicates what types of events are to be audited and can be set on a per-system, per-project, or per-user basis. Post-audit reduction tools (the print sys log and summarize sys log commands) provide the capability to print audit records that match a given format or that contain a particular string, and to generate reports summarizing those records.

Multics provides a mechanism that is able to monitor the occurrence or accumulation of security auditable events that may indicate an imminent violation of security policy. This mechanism is able to immediately notify the security administrator when thresholds are exceeded. This capability is provided by the monitor sys log command, which can execute an arbitrary command line immediately after a specified (by type or contents) message appears in the log. Additionally, some logged events always generate an audible alarm at the system console, such as too many bad login attempts or an attempt to use a locked password. Thus, Multics provides a capability, real-time alarms, which exceeds the auditing requirements for a class B2 system.

Conclusion

Multics MR11.0 satisfies the B2 Audit requirement.

System Architecture

Requirement

> The TCB shall maintain a domain for its own execution that protects it from external interference or tampering (e.g., by modification of its code or data structures). The TCB shall maintain process isolation through the provision of distinct address spaces under

> its control. The TCB shall be internally structured into well-defined largely independent modules. It shall make effective use of available hardware to separate those elements that are protection-critical from those that are not. The TCB modules shall be designed such that the principle of least privilege is enforced. Features in hardware, such as segmentation, shall be used to support logically distinct storage objects with separate attributes (namely: readable, writeable). The user interface to the TCB shall be completely defined and all elements of the TCB identified.

Applicable Features

The Multics TCB consists of three major parts: ring zero, ring one, and the privileged processes. Ring zero and ring one provide the bulk (hundreds) of the user-invocable functions. Privileged processes primarily provide internal TCB functions; they supply only a small number of user-invocable functions. The ring zero and ring one portions are isolated from user action by the hardware ring and segment access control mechanisms. The privileged processes are isolated by the fundamental separation between processes and by the DAC mechanism. The TCB uses hardware segmentation to maintain a completely distinct address space for each process; this is performed in ring zero. Each process has (potentially) a different set of segments to which it has access, and (potentially) different access rights to those segments.

Only the ring zero portion of the TCB has direct access to the data structures defining the process address space, process isolation, and file system objects, so these cannot be affected by other parts of the TCB. Additionally, only a small number of ring zero TCB programs are permitted to execute privileged instructions. Similarly, ring one data structures cannot be directly affected by most privileged processes, and privileged processes cannot be directly affected by other privileged processes or the ring one TCB. All these isolations are provided by the hardware separation of process address spaces and per-segment access control (ring brackets and access modes) in those address spaces. This effectively separates the most protection-critical components of the TCB from the rest.

The TCB, written primarily in PL/I, is well-structured, and isolates functions and data structure manipulations into separate modules (see page 16, "Software Architecture"). Most access decisions are made in per object access kernel programs, and all

use the same two basic primitives to interpret DAC and MAC data structures. Because access to data structures is well-modularized, and the TCB is separated into hardware-isolated components, use of privilege is minimized. The user interface to the TCB is completely defined by the DTLS, which describes all user-callable entrypoints into ring zero and ring one and defines the protocols for communicating with the privileged processes.

Conclusion

Multics MR11.0 satisfies the B2 System Architecture requirement.

System Integrity

Requirement

> Hardware and/or software features shall be provided that can be used to periodically validate the correct operation of the on-site hardware and firmware elements of the TCB.

Applicable Features

A variety of tests are available to verify correct functioning of Multics hardware and firmware. Together, these tests are known as the Test and Diagnostic (T&D) software. Most T&D software (CPU, MPC, FNP, and peripheral tests) can be run while the system is running, by isolating the component under test from the rest of the system and testing it under control of the Multics TCB. All T&D software can be run when the system is not operating, by dedicating the hardware to running T&D.

In addition to the T&D software, two classes of tests exercise the system under load in order to identify transient failures. The first of these is called MHAT (Multics Hardware Acceptance Tests); it creates a heavy load with a variety of applications. The other consists of a set of tests for specific instructions that have been found subject to transient failures in the past. Both of these additional tests can be run without any dedicated hardware.

Conclusion

Multics MR11.0 satisfies the B2 System Integrity requirement.

Covert Channel Analysis

Requirement

> The system developer shall conduct a thorough search for covert storage channels and make a determination (either by actual measurement or by engineering estimation) of the maximum bandwidth of each identified channel.

Applicable Features

The Covert Channel Analysis(1) provided by the vendor identified 47 potential covert channels. Of these, 37 were covert storage channels, 4 were covert timing channels, and 6 were not covert channels at all. The analysis provided engineering estimates of the maximum bandwidths for all the identified channels. The storage channels were categorized as follows:

Description	Bandwidth	Number Identified	Disposition
large	› 100 BPS	10	Moved to a lower category by adding noise or closed completely
moderate	10 - 100 BPS	8	Audited or moved to a lower category
small	1 - 10 BPS	5	Audited or documented
trivial	‹ 1 BPS	11	Ignored
privileged		2	Documented
generic		1	Closed completely

Conclusion

Multics MR11.0 satisfies the B2 Covert Channel Analysis requirement.

---

(1) Multics Covert Channel Analysis, Honeywell Information Systems, Inc., May 1985 (Honeywell Proprietary).

Trusted Facility Management

Requirement

> The TCB shall support separate operator and administrator functions.

Applicable Features

Multics supports separate operator and administrator functions by restricting operators to the commands required to operate the system, and by requiring administrative functions to be performed by administrators either in their own processes or from a system console after supplying a special password. No operator commands can change the security attributes of any protected objects. Both operators and administrators must login to the system before performing their functions. Administrators login as privileged user proceses, on normal consoles. Operators login through an equivalent mechanism, but can only do so when using designated operator consoles. Operator login goes through similar authentication procedures as user login, but rather than creating a process, simply identifies the operator to the system control process.

Conclusion

Multics MR11.0 satisfies the B2 Trusted Facility Management requirement.

Security Testing

Requirement

> The security mechanisms of the ADP system shall be tested and found to work as claimed in the system documentation. A team of individuals who thoroughly understand the specific implementation of the TCB shall subject its design documentation, source code, and object code to thorough analysis and testing. Their objectives shall be: to uncover all design and implementation flaws that would permit a subject external to the TCB to read, change, or delete data normally denied under the mandatory or discretionary

> security policy enforced by the TCB; as well as to assure that no subject (without authorization to do so) is able to cause the TCB to enter a state such that it is unable to respond to communications initiated by other users. The TCB shall be found relatively resistant to penetration. All discovered flaws shall be corrected and the TCB retested to demonstrate that they have been eliminated and that new flaws have not been introduced. Testing shall demonstrate that the TCB implementation is consistent with the descriptive top-level specification.

Applicable Features

The security mechanisms of Multics were tested by penetration and functional testing (for more detail see page 93, "Functional Testing"). Penetration testing took place from February to July, 1984. Penetration testing began with flaw hypothesis generation which was partially based on examination of design documentation and source code. Investigation of the flaw hypotheses involved inspecting and analyzing source code. All flaws identified were implementation errors. No design errors were found. Few flaws were found, and those found were corrected in MR11.0. These corrections were examined and introduced no new problems.

Functional testing occurred in two phases. In the first phase, the team developed tests which exercise some of the user-visible TCB interfaces. Semi-automated tests were used to exercise the TCB interfaces that enforce discretionary access controls, mandatory access controls (AIM), and labeling. Identification and authentication, auditing, and object reuse were tested using manual techniques.

During the second phase of functional testing, the evaluation team reviewed and executed the functional test suite provided by Honeywell. This test suite, along with the tests developed by the team, consists of over 64,000 lines of code which test approximately 700 different gate entry points. The results of executing the entire test suite on MR11.0 showed that the Multics security-related mechanisms used by non-privileged user interfaces work as documented to enforce the security policy.

Conclusion

Multics MR11.0 satisfies the B2 Security Testing requirement.

Design Specification and Verification

Requirement

> A formal model of the security policy supported by the TCB shall be maintained that is proven consistent with its axioms. A descriptive top-level specification (DTLS) of the TCB shall be maintained that completely and accurately describes the TCB in terms of exceptions, error messages, and effects. It shall be shown to be an accurate description of the TCB interface.

Applicable Features

The Bell and LaPadula model(1) is the formal model of Multics security policy. "Multics Security Model -- Bell and LaPadula"(2) is Honeywell's interpretation of the Bell and LaPadula model for Multics. This document correlates the subjects, objects, and access modes in the Bell and LaPadula model with those implemented by Multics. The Multics DTLS is considered to be the Multics Subroutines and I/O Modules(3) manual. This manual is a complete and accurate description of the user interface to the TCB in terms of exceptions, error messages, and effects.

Conclusion

Multics MR11.0 satisfies the B2 Design Specification and Verification requirement.

---

(1) Bell, D. E. and LaPadula, L. J., Secure Computer Systems: Unified Exposition and Multics Interpretation. MITRE Corp., Bedford, MA., MTR-2997, July 1976.

(2) Tague, R. M. Multics Security Model -- Bell and LaPadula. Honeywell Information Systems, Inc., MDD-002, August 1985.

(3) Multics Subroutines and I/O Modules. Honeywell Information Systems, Inc., AG93-05, May 1985.

Configuration Management

Requirement

> During development and maintenance of the TCB, a configuration management system shall be in place that maintains control of changes to the descriptive top-level specification, other design data, implementation documentation, source code, the running version of the object code, and test fixtures and documentation. The configuration management system shall assure a consistent mapping among all documentation and code associated with the current version of the TCB. Tools shall be provided for generation of a new version of the TCB from source code. Also available shall be tools for comparing a newly generated version with the previous TCB version in order to ascertain that only the intended changes have been made in the code that will actually be used as the new version of the TCB.

Applicable Features

The Multics configuration management system maintains control of changes to the DTLS (see page 85, "Design Specification and Verification"), the Multics Design Documents (MDDs), source code, object code, change histories, functional test suites, and change information. It assures that all these items are mapped to the current version of the TCB. Multics provides tools to generate and install the current version of the TCB from source code and to retrieve relevant previous versions. Multics provides online forums for reviewing suggested design changes. A history comment tool assures the correct recording of the change history of a source module. Online critical fix libraries record critical changes that have occurred since a release. An ASCII compare tool allows identification of exactly what changes have been made to a module. A tracking database allows tracking of changes to the TCB.

These tools and many other supporting installation tools along with the procedures that comprise the configuration management system are detailed in the following Multics documents:

MAB-070	Multics Configuration Management: Policy Statement
MAB-066	Multics Configuration Management: Software Development
MAB-068	Configuration Management: Programming Standards
MAB-048	Rules for the Multics Change Review Board

MAB-069	Multios Configuration Management: Guidelines for Auditing Software
MAB-071	Multios Security Coordinator: Duties and Responsibilities
MAB-067	Procedures for Software Installation and Integration
MAB-056	Multics Configuration Management: Installing Planned Changes in System Libraries
MAB-057	Multios Configuration Management: Installing Emergency Changes in System Libraries
MAB-063	Multics Configuration Management: Critical Fixes for Released Software
MTB-716	Multics Configuration Management: Tracking Software Changes for MR12.0
MDD-004	The Multics Security Functional Test Suite: Goals, Standards and Policies

Conclusion

Multics MR11.0 satisfies the B2 Configuration Management requirement.

Security Features User's Guide

Requirement

> A single summary, chapter, or manual in user documentation shall describe the protection mechanisms provided by the TCB, guidelines on their use, and how they interact with one another.

Applicable Features

Chapter 6 of the Multics Programmers' Reference Manual (AG91-04) describes the security mechanisms available on Multics, which are User Name/Password, Access Control Lists, Access Isolation Mechanism, and ring mechanism. This document contains guidelines on the use of the security mechanisms, and a description of their interaction.

Conclusion

Multics MR11.0 satisfies the B2 Security Features User's Guide requirement.

Trusted Facility Manual

Requirement

> A manual addressed to the ADP system administrator shall present cautions about functions and privileges that should be controlled when running a secure facility. The procedures for examining and maintaining the audit files as well as the detailed audit record structure for each type of audit event shall be given. The manual shall describe the operator and administrator functions related to security, to include changing the security characteristics of a user. It shall provide guidelines on the consistent and effective use of the protection features of the system, how they interact, how to securely generate a new TCB, and facility procedures, warnings, and privileges that need to be controlled in order to operate the facility in a secure manner. The TCB modules that contain the reference validation mechanism shall be identified. The procedures for secure generation of a new TCB from source after modification of any modules in the TCB shall be described.

Applicable Features

The information required for a trusted facility manual is contained in Part VI (Assuring System Security) and Appendix B (Audit Tables and Include Files) of the Multics System Administration Procedures Manual, May 1985 (AK50-03). Part VI consists of chapters 18 through 26 of the manual and provides guidelines for the system administrator on how to manage Multics as a secure system. Chapters and chapter headings are:

Chapter 18 Assuring the Security of the File System
Chapter 19 Assuring the Security of RCPRM
Chapter 20 Communication Channel Security
Chapter 21 Assuring the Security of the I/O Daemons
Chapter 22 Absentee Facility Security
Chapter 23 Privileged Operations Security
Chapter 24 System Logs
Chapter 25 Security Auditing
Chapter 26 Miscellaneous Security Tasks

Operator and Administrator functions related to security are addressed throughout Part VI. Guidelines on the consistent and effective use of the protection features of the system and interaction of the protection features also occur throughout Part VI, as do facility procedures, warnings, and privileges that need to be controlled in order to operate the facility in a secure manner.

Procedures for examining and maintaining the audit files are contained in Chapter 25 (Security Auditing). Chapter 25 and Appendix B contain the detailed audit record structure for each type of audit event.

Instructions for changing seourity characteristics of a user appear in Chapter 18, page 18-31, under the headings "Specifying Authorizations for Users" and "Specifying Authorizations for Projects". Additional instructions are given in Chapter 23, page 23-2, under the heading "Granting Access to Use System Privileges".

The TCB modules that contain the reference validation mechanism are described in Chapter 26 (Miscellaneous Security Tasks), page 26-9, under the heading "Reference Monitor Implementation".

Guidelines on how to securely generate a new TCB are described in Chapter 26 (Miscellaneous Security Tasks), page 26-5, under the heading "Reviewing Software Changes in New Software Releases". Additional guidelines are included in the Multics Installation Bulletin, which accompanies each new release. The procedures for secure generation of a new TCB from source after modification of any modules in the TCB are essentially no different than those used for generation of a new TCB. TCB Modules installed individually as a result of a critical fix are installed via standard Multics commands which are documented in the Multics Commands and Active Functions Manual (AG92-06).

Conclusion

Multics MR11.0 satisfies the B2 Trusted Facility Manual requirement.

Test Documentation

Requirement

> The system developer shall provide to the evaluators a document that describes the test plan and results of the security mechanisms' functional testing. It shall include results of testing the effectiveness of the methods used to reduce covert channel bandwidths.

Applicable Features

Honeywell made the results of their functional test suite available to the evaluation team. Also available for review were documentation describing existing tests, the procedures for creation of new functional tests, and the procedures for execution of the existing test suite. The team reviewed the test suite and found very good security coverage. The documentation was reviewed and found to be sufficiently detailed and accurate.

The covert channel analysis documentation provided by the vendor included the results of testing the methods used to reduce covert channel bandwidths.(1) The documentation was examined by the team to ensure that the estimated bandwidths of channels were accurate, that the methods used to reduce bandwidths were actually effective, and that the reduction methods did not introduce additional security-related problems.(2)

Conclusion

Multics MR11.0 satisfies the B2 Test Documentation requirement.

---

(1) Loepere, Keith, Resolving Covert Channels within a B2 Class Secure System, Operating Systems Review, Vol. 19, Number 3, July 1985.

(2) Multics Covert Channel Analysis, Honeywell Information Systems, Inc., May 1985 (Honeywell Proprietary).

Design Documentation

Requirement

> Doocumentation shall be available that provides a description of the manufacturer's philosophy of protection and an explanation of how this philosophy is translated into the TCB. The interfaces between the TCB modules shall be described. A formal description of the security model enforced by the TCB shall be available and proven that it is sufficient to enforce the security policy. The specific TCB protection mechanisms shall be identified and an explanation given to show that they satisfy the model. The descriptive top-level specification (DTLS) shall be shown to be an accurate description of the TCB interface. Documentation shall desoribe how the TCB implements the reference monitor concept and give an explanation why it is tamperproof, cannot be bypassed, and is correctly implemented. Documentation shall describe how the TCB is structured to facilitate testing and to enforce least privilege. This documentation shall also present the results of the covert channels analysis and the tradeoffs involved in restricting the channels. All auditable events that may be used in the exploitation of known covert storage channels shall be identified. The bandwidths of known covert storage channels, the use of which is not detectable by the auditing mechanism, shall be provided.

Applicable Features

The Multics interpretation of the Bell and LaPadula model(1) provides a description of Honeywell's philosophy for protection and how this is translated into the TCB. The security model enforced by the TCB is the Bell and LaPadul model,(2) as described on page 85, "Design Specification and Verification".

---

(1) Tague, R. M. Multics Security Model -- Bell and LaPadula. Honeywell Information Systems, Inc., MDD-002, August 1985.

(2) Bell, D. E. and LaPadula, L. J., Secure Computer Systems: Unified Exposition and Multics Interpretation. MTR-2997, The MITRE Corp., Bedford, MA., July 1976.

Multics has a set of Multics Design Documents (MDDs)(1) that describe the TCB. (These documents are Honeywell Internal Documentation and are available only through the vendor by request. Honeywell reserves the right to deny such requests.) The MDDs are organized by major TCB service or function. These design documents describe the interfaces between TCB modules, how the TCB implements the reference monitor, and how the TCB is structured to facilitate testing and enforce least privilege. These documents coupled with the Honeywell-produced Multics Interpretation referenced in the previous paragraph identify the security protection mechanisms and explain how they satisfy the model.

The DTLS as described on page 85, "Design Specification and Verification", is an accurate description of the TCB interface.

The covert channel analysis describes all identified covert channels, how they can and cannot be restricted, how they are audited, and their bandwidths. For a more detailed description, see page 82, "Covert Channel Analysis".

Conclusion

Multics MR11.0 satisfies the B2 Design Documentation requirement.

---

(1) Currently the MDD set is not complete and for this evaluation was supplemented by discussions with the developers and comments in the code. The remaining documents needed to complete the set will be available for Multics MR12.0.

SOFTWARE TESTING

Functional Testing

The functional tests developed by Honeywell were both manual and automated. The discretionary access control, mandatory access control (AIM), audit, and labeling mechanisms were tested using automated test suites. The identification and authentication mechanisms were tested using manual test scripts.

Honeywell's automated tests took the form of PL/I programs. These exercised the TCB through the non-privileged user accessible interfaces called gates (see page 11, "Gates and Ring Changing"). Numerous test cases were defined for each gate entry to assure that the entry does not violate the security properties. Each test case defines the input parameters to the entry, the calling environment and the expected results. The programs setup the environment, call the entry, then compare the actual with the expected results.

Special utilities were constructed to allow the test programs to automatically sequence through a central table of cases for the discretionary and mandatory controls. Also, the test suite was developed under strict configuration management control. This contributed to very uniform implementation and test coverage.

Honeywell's manual tests consisted of fourteen scripts used to test identification and authentication mechanisms. The scripts were designed to drive the Answering Service through approximately 280 login attempts that exercise the entrance paths to Multics.

Honeywell's tests were executed individually on the CISL-Service Multics system. They were then integrated into a few large suites and executed on a Multics test system in Phoenix, AZ.

The functional tests developed by the team were both manual and semi-automated. The discretionary access control, labeling, and mandatory access control (AIM) mechanisms were tested using semi-automated test suites. The other security-relevant mechanisms of the system were exercised using manual tests.

The semi-automated tests took the form of command files which, when executed, created output segments. These output segments were saved for manual inspection or automatically compared to

previously validated functional test outputs. The previously validated outputs were online and available for manual inspection.

The team's manual tests were a series of calls at command level to security mechanisms. The expected results (which were determined beforehand) and the actual results returned by the system were then compared.

The team's test suite was executed on two other Multics systems, the U. S. Air Force's (1st Information Services Group) System-T and Honeywell's CISL-Service system, to validate accuracy and portability.

Honeywell also extended the team's semi-automatic tests to include testing of the auditing mechanism. (In the future, Honeywell expects to convert all of the team's tests to the automatic PL/I program approach. Until then, Honeywell will maintain these tests in their current form.)

Development and execution of the test suite exposed approximately 72 flaws in MR11.0. Most of these are not exploitable in any meaningful fashion. Honeywell corrected the remainder in a set of critical fixes. Customer sites are expected to install a package containing these changes.

The evaluation team developed and executed approximately 145 functional tests which covered 20-25% of the non-privileged (user accessible) TCB operations. Honeywell developed approximately 385 tests to attain complete coverage of the non-privileged TCB operations. Execution of the entire test suite took place during the summer of 1985 on Honeywell's System-M in Phoenix, AZ. It is felt that the test suite provides good coverage in exercising the system security properties.

Penetration Testing

Penetration testing of Multics was a three month effort. The team prepared for the effort by reading a variety of papers describing previous penetration efforts. Team members had also collected a list of possible flaw areas during Multics training, while mapping the system to the Criteria and while completing the functional testing.

The actual penetration effort began with flaw hypothesis generation. Forms were set up to record flaws, and approximately 75 possible flaws were suggested in the initial attempt. Each

flaw was then assigned "probability of successful exploitation" and "ease of determining existence" factors. Those flaws which were considered easy to test and likely to exist were explored first. Exploration of flaws often resulted in new hypotheses being added to the list. Sometimes an exploited flaw would be a specific instance of a generic flaw.

Work on exploring the flaw hypotheses continued for three months elapsed time. The team worked in physical proximity, with access to multiple terminals in the same room at the Pentagon for one third of that time. The other two months were spent working in the team members' normal working environment.

Test Results

A total of 106 hypotheses were generated over the course of the penetration testing. Of these, 84 were explored, and 70 were confirmed as flaws. Most of the flaws (75%) were not exploitable in any meaningful fashion. No flaws were design errors; all were implementation errors, mostly simple programming errors. Honeywell corrected all non-trivial flaws in Multics MR11.0.

This page intentionally left blank.

EVALUATORS' COMMENTS

Overall Impressions

The evaluation team believes that Multics provides an excellent general-purpose time-sharing environment for the typical user. It is a technically sound, functionally mature operating system, that has been used in large-scale software development, relational database applications, network-oriented applications, and applications requiring unique security constraints. Multics fully supports the lattice model of the DoD security policy. Multics is particularly strong in its online help facilities, program development environment, interactive conference facilities and secure electronic mail.

Multics provides security by methods which are designed into and are an integral part of the functioning of the system. Through this design, security benefits are achieved with insignificant system overhead. The hardware-supported protection rings and segmentation provide tightly controlled separate domains of execution. The Access Isolation Mechanism (AIM) software provides mandatory access control. The Access Control Lists (ACLs) provide discretionary access control. Any change in the protection attributes of objects takes effect immediately. This is unique to the team's experience.

Protected Subsystems

Multics uses a hardware-implemented multilevel ring structure to control its users and to protect itself. The ring structure is a generalization of the two-state capability of other computer systems, but has been expanded to eight states or "rings of protection".

The use of eight hierarchical levels of protection rather than two allows application programs to take advantage of protection features normally reserved for operating system software. Development of protected subsystems is not only useful, but extremely easy to implement. For example, one could set up a database that could be accessed only from specific programs, thereby allowing the type of finely grained access control necessary for many applications. This mechanism is used by

standard Multics products, such as the Forum interactive meeting system, and is also used by Multics customers for their own applications.

The use of the ring structure is historically and technically significant. The Multics Operating System is frequently referenced in Computer Science textbooks on operating system theory. Multics was the first to develop and implement a product with a multilevel ring structure and has served as a model for many other computer systems, including most of the important concepts in UNIX.(1)

Reconfiguration

Multics is a large-scale multiprocessor system capable of delivering effective performance from a large number of CPUs; it is limited to only six by the hardware implementation, not the operating system software. Additionally, almost all hardware components in the system can be disconnected under software control during normal system operation and later brought back into service. For instance, a CPU or I/O device can be logically deleted, tested, and brought back into operation without affecting the system except for temporarily reduced performance.

Denial of Service

Denial of service occurs through system "crashes" or when the system is not available to the user because it is pre-empted by systems programmers. While not an issue of the Criteria, reliability is an important consideration in judging the usefulness of a computer system.

In Multics, most system crashes are caused by hardware or environmental (power, air conditioning) problems. Due to the security-oriented system architecture and the extensive structured software testing procedures, it is very rare for a software problem to allow a user to cause a system crash. The operating system isolates damage from runaway programs through its implementation of a two-dimensional address space. When system crashes do occur, the system may be rebooted

---

(1) UNIX is a registered trademark of AT&T Bell Laboratories, Inc.

automatically, even in the unattended operation mode. To prevent severe loss of data in the event of a system crash, the system continuously backs up all modified files on tape. Scheduled system down time is held to a bare minimum because of Multics' ability to perform online diagnosis, testing, maintenance, and software installation. Additionally, the system can detect damage and, in most cases, repair it automatically.

This page intentionally left blank.

EVALUATED HARDWARE COMPONENTS

Scope of Hardware Evaluation

The hardware covered by this evaluation is the entire current Multics product line, including supported hardware present in the field at existing customer sites.

Although much of this hardware was not physically inspected or tested, the evaluation team examined engineering specifications for all hardware components to assure themselves that the components were equivalent, or where not equivalent, made no security-relevant differences in the TCB. This examination process included interviews with Honeywell engineers during which the team and Honeywell focused on the differences between hardware components and evaluated their impact on the TCB. Overall, very little code in the TCB deals with different types of hardware, and the differences between different models of hardware components are insignificant.

The primary requirement for hardware evaluation is that the hardware function properly. This was verified by the system integrity tests (see page 81, "System Integrity") and was not given a detailed reevaluation by the team.(1) The integrity assurances provided by the Honeywell-supplied diagnostic tests are satisfactory.

List of Evaluated Components

This section lists, in several categories, the Honeywell marketing identification numbers for all hardware covered by this evaluation. This includes both hardware currently in production and some, but not all, Multics hardware that was produced in the past. This list is equivalent to the set of hardware officially supported by the evaluated release and includes all hardware currently known to be in use at customer sites.

---

(1) Some CPU and MPC protection mechanisms were explored during the penetration testing, but no attempt was made to do exhaustive testing. In any case, no problems were found.

Although peripherals were not evaluated as such, the system was evaluated only for running with the supported set of peripherals. Therefore, they are included in this list to allow a determination that a particular configuration contains only devices that are supported by MR11.0.

In cases where the detailed description of options or devices is not interesting from a security standpoint, the marketing identifiers have simply been listed without descriptions.

To operate in correspondence with the B2 rating, the hardware configuration must contain only components listed in this section.

Central System Combinations

Multics systems are typically sold as a "central system" plus additional components. The central system is merely a bundling of hardware components (CPUs, SCUs, and IOMs) listed below.

CPS8802	Level 68/DPS-1 Central System
CPS8803	Level 68/DPS-M Central System
CPS8193	DPS 8/52M Central System
CPS8194	DPS 8/62M Central System
CPS8199	DPS 8/70M Central System

Additional CPUs and CPU Enhancements

This list covers all the CPU models (as purchased individually) and the field-installable performance upgrades for converting one CPU model to another or enhancing the performance of a particular CPU. The Level 68 hardware in this list, and in the central systems list above, does not include all the marketing identifiers ever used for that hardware; however, field upgrades to existing processors have made them all equivalent to models listed below.

CPK8004	Performance Addition for CPS8002 (Level 68 Cache)
CPU8803	Additional Level 68 Processor for CPS8002 and CPS8003
CPU8193	DPS 8/52M Additional Processor
CPU8194	DPS 8/62M Additional Processor
CPU8199	DPS 8/70M Additional Processor
CPK8194	Upgrade Kit - 8/52M to 8/62M Central System

CPK8195	Upgrade Kit - 8/52M to 8/62M Additional Processor
CPK8197	Upgrade Kit - 8/62M to 8/70M Central System
CPK8198	Upgrade Kit - 8/62M to 8/70M Additional Processor
CPF8199	Performance Enhancement Feature (DPS-8/70M 32K Cache)
RSF8003	Redundant Systems Feature

Additional IOMs and IOM Enhancements

MXU6002	Additional Freestanding IOM
MXF6005	IOM Expansion - 35 to 54 Board Slots
MXU8002	Additional IOM
MXP8005	IOM Channel Expansion (36 to 54 slots)
MXF8012	IOM Logical Channel Expansion (24-56 channels)

Additional SCUs and Memory

MXC6004	Additional Freestanding SCU
MXC8001	Additional Freestanding SCU
MXC8002	Additional Systems Controller
CMA6050 to CMA6061	Memory addressing features for MXC6004
CMM6050 to CMM6061	Additional Memory for MXC6004
CMM8011 to CMM8016	Additional Memory for MXC8001
CMM8020	Additional 2MB Main Memory for MXC8002

Disk MPCs and Options

MSP0601	Freestanding Mass Storage Processor
MSP0603	Mass Storage Processor
MSP0607	Freestanding Single Channel MSP
MSP0609	Freestanding Dual Channel MSP
MSK6007	Upgrade kit - MSP0607 to MSP0609
MSP0611	Single Channel Mass Store Processor
MSP0612	Dual Channel Mass Store Processor
MSP8021	Freestanding Primary Mass Store Processor
MSP8022	Integrated Secondary Mass Store Processor
MSP8023	Integrated Primary Mass Store Processor

Miscellaneous Disk MPC addressing and interconnect options:

MSF1021, MSF1025, MSF1045, MSF1028, MSA1027, MSA1030, MSA1029, MSF1023, MSF1035, MSF1024, MSF1033, MSF1031, MSF1036, MSA1042, MSA1043, MSF1041, MSF1042, MSF1140, MSK0612, MSA1140, MSA1141, PSS8001, MSA1142, MSA1143, MSF1141, MSF1142, MSK1141, MSK1142, MSK1143, MSF8021, MSF8022, MSA8011, MSF8012

Tape MPCs and Options

MTP0610	Magnetic Tape Processor
MTP0611	Magnetic Tape Processor
MTP8021	Freestanding Primary Magnetic Tape Processor
MTP8022	Integrated Secondary Magnetic Tape Processor
MTP8023	Integrated Primary Magnetic Tape Processor

Miscellaneous Tape MPC addressing and interconnect options:

MTA1142, MTF1141, MTF1145, MTF1146, MTF1147, MTF1148, MTF1149, MTF1150, MTA1152, MTF1151, MTF1155, MTF1156, MTF1157, MTF1158, MTF1159, MTF1160, MTF8021, MTF8022, MTA8011, MTF8012, MTF8023, MTF8024, PSS8202, PSS8203

Unit Record MPCs and Options

URP8001	Unit Record Processor (2 Card Units & 2 Printers)
URP8002	Unit Record Processor (2 Card Units)
URP8003	Unit Record Processor (2 Printers)
URP8004	Unit Record Processor (2 PRU0901/1201 or PRU1200/1600 Printers)

Communication Processors and Options

There is only one supported model of communications processor (FNP), but it has in the past been known by other marketing identifiers (such as DCF6652 and DCF6678) identifying various bundled assemblies. All extant FNPs, however, are equivalent to systems assembled from components listed below. The communication channel options supported by the FNP have not been evaluated specifically, since they contain no security-relevant features of their own; they are listed here for informational purposes only.

DCU6661	DATANET Communications Processor

DCE6661	Performance and Connectivity Enhancement
DCE6662	Cache Memory Performance Enhancement
DCE6663	Additional Line Configurability
DCE6605 to DCE6607	Additional Memory (64 to 512 KB)

DCF6607	Channel Interface Base
DCF6610	Dual Channel Paokage - Current Loop
DCF6611	Dual Channel Package - Synchronous RS-232-C
DCF6612	Dual Channel Package - Asynchronous RS-232-C
DCF6613	Automatic Call Unit - Dual Channel
DCF6614	Synchronous Channel - MIL STD
DCF6615	Dual Channel Package - Asynchronous MIL STD
DCF6616	Broadband Channel - MIL STD
DCF6617	HDLC Channel - MIL STD
DCF6618	Dual Channel Paokage - Binary Synchronous
DCF6619	Broadband Channel
DCF6620	HDLC Voice Grade Channel
DCF6621	Broadband Channel - Binary Synchronous
DCF6622	HDLC Broadband Channel
DCF6623	HDLC Broadband Channel - V.35
DCF6627	Broadband Channel - V.35
DCF6626	Direct Connect Capability - RS-232-C
DCF6927	Universal Modem Bypass

Peripheral Equipment (not evaluated)

This section lists all the peripheral devices supported by MR11.0, and their options. These devices have not been evaluated specifically, since they contain no security-relevant components.

Disk Storage Devices and Options:
MSU0451, MSU0500, MSF0006, MSF0007, MSF0011, MSU0501, MSK0501

Magnetic Tape Drives and Options:
MTU0610, MTF0607, MTF0608, MTK0678, MTU0630, MTF0634, MTF0635, MTF0636, MTF0637, MTK0630, MTK0631, MTK0632, MTK0633, MTK0634

Unit Record (Printer, Card Reader, Card Punch) Equipment and Options:
PRU1200, PRU1600, PRK1216, PRB0600, PRU0901, PRU1201, PRF0045, PRK0901, PRB3300, PRB3600, PRU0903, PRU1203, PRK0903, CRU1050, PCU0121, CRU0501, CRF0030

System Console Equipment and Options:
CSU6601, CSF6601, CSU8001

This page intentionally left blank.

EVALUATED SOFTWARE COMPONENTS

Scope of Software Evaluation

This section lists the programs that make up the various major divisions of Multics software. Each subsection describes the division and lists the programs contained therein. These groupings are only approximate, since any particular program may be used in several different environments.

Multics Central System TCB

In normal operation, the Multics Central System Trusted Computing Base comprises approximately 635,000 lines of code, principally in PL/I with a small portion in assembler language. Additional code is used for initialization, in the Front-end Network Processors, and in the Microprogrammed Controllers.

Ring Zero TCB

Approximately 149,000 lines of code. This represents all functions performed only in ring zero after system initialization is completed. It includes functions such as page control, the scheduler, the file system, and the communications system.

bound_355_wired
bound_active_1
bound_dir_control
bound_disk_util_1
bound_disk_util_2
bound_error_active_1
bound_error_active_2
bound_error_wired_2
bound_file_system
bound_hc_backup
bound_hc_data_wired
bound_hc_reconfig
bound_hc_tuning
bound_interceptors
bound_io_active
bound_io_wired
bound_iom_support
bound_tc_wired
bound_tty_active
bound_unencacheable
bound_vtoc_man
bound_wired_1
bound_x25_mpx
emergency_shutdown
hasp_mpx
ibm3270_mpx
init_processor
polled_vip_mpx
restart_fault
signaller
active_all_rings_data
active_hardcore_data
ast_look_meter_seg
inzr_stk0

bound_mcs_util
bound_mdir_control
bound_page_control
bound_priv_1
bound_priv_mpx
bound_process_creation
bound_salvager
bound_scavenger
bound_segment_control
bound_system_security
bound_tc_priv
pds
prds
pvt
salv_data
sst_seg
sys_info
syserr_data
tc_data
template_pit
tty_buf

Ring One TCB

Approximately 60,000 lines of code. This represents all TCB functions performed in ring one, on behalf of both normal and TCB processes. It includes RCP, message segments, and some miscellaneous functions.

bound_mdxhdx_
bound_ms_table_mgr_
bound_mseg_
bound_mseg_old_
bound_pnt_interface_
bound_rcp_
bound_rcprm_
bound_ssu_
bound_tape_label_util_
default_rtmf
ring_zero_peek_filter_

TCB Support Code

Approximately 158,000 lines of code. This represents all the runtime support functions used by all processes and in all rings. It includes functions such as PL/I language runtime support, process initialization, conversion routines, the process I/O system, and the basic process environment for TCB processes. It does not include runtime support for other languages, since they are not used within the TCB.

bound_command_env_
bound_command_loop_
bound_conversion_rtns_
bound_date_time_
bound_debug_util_
bound_error_handlers_
bound_expand_path_
bound_fs_util_
bound_fsim_
bound_full_cp_
bound_ios_
bound_log_support_
bound_pl1_runtime_
bound_pli02_
bound_process_env_
bound_process_init_
bound_qedx_
bound_search_facility_
bound_segment_info_
bound_tape_mult_
bound_ti_term_
bound_vfile_

bound_ipc_
bound_library_1_
bound_library_2_
bound_library_3_
bound_library_wired_
bound_log_active_
error_table_
free_
operator_pointers_
search_list_defaults_
trace_operator_pointers_

Dedicated TCB Processes

Approximately 177,000 lines of code. This represents all funotions required by the standard TCB processes (see page 24, "Dedicated TCB Processes"). It includes such software as the Answering Service, line printer support, and the dumpers and reloaders.

as_error_table_
bcd_prt_image
bisync_
bound_absentee_ctl_
bound_as_install_ctl_
bound_as_mc_
bound_as_misc_
bound_as_mpx_
bound_as_requests_
bound_as_user_message_
bound_card_dims_
bound_card_input_
bound_daemon_ctl_
bound_dump_tape_
bound_dumper_
bound_exec_com_
bound_g115_
bound_hasp_
bound_hc_initlzr_auxl_
bound_ibm3270_
bound_iodc_
bound_iodd_
bound_lss_
bound_mcs_init_
bound_misc_io_modules_
bound_oc_
bound_print_sheets_
bound_prtdim_
bound_rcp_op_cmnds_
bound_reloader_
bound_remote_io_
bound_system_control_
bound_system_startup_
bound_user_ctl_
bound_volume_bk_tools_
bound_volume_dumper_
bound_volume_reloader_
bound_volume_retv_
bound_volume_rldr_ut_
bound_word_generator_
gm_path_list
hierarchy_backup_dumper_lss
volume_dumper
volume_dumper_lss
volume_reloader
volume_reloader_lss
volume_retriever
volume_retriever_lss

System Administration

Approximately 96,000 lines of code. This represents the functions used by the system adminstrator, accounting administrator, system security officer, and system maintenance

personnel. It includes such things as administrative table installation, audit log perusal, hardware maintenance, and user-ID creation.

bound_admin_billing_
bound_admin_old_
bound_admin_rtnes_
bound_admin_tools_
bound_dn355_tools_
bound_dpu_support_
bound_install_table_
bound_io_tools_
bound_log_tools_
bound_priv_commands_
bound_priv_rtnes_
bound_proj_admin_
bound_security_tools_
bound_tolts_
copy_dump
initialize_peek_limits
tolts_overseer_

System Initialization

Approximately 73,000 lines of code. This represents the mechanisms used to generate a new Multics TCB and establish a secure state during system initialization. It includes the Bootload Operating System (BOS), the Bootload Command Environment (BCE), and the system generation tools.

abs
apnd
blast
boot
bootload_1
bootload_tape_label
bostap
bound_bce_dump_
bound_bce_exec_com_
bound_bce_paged
bound_bce_probe_
bound_bce_wired
bound_bootload_0
bound_checker_
bound_gm_
bound_init_1
bound_init_2
bound_io_init
bound_multics_bce_
bound_temp_1
bound_temp_2
config
core
die
dmp355
dump
fmt
fwload
if
label
ld355
loaddm
mpcd
mst_boot_label
ncopy
ndisk
ntape
p1200i
p300i
patch
print
ptpkg
rdlabl
restor
runcom
salv
save
setup
sstn
taped
test
time

edit	tst3bt
esd	tstchn
fd355	util
fdump	write

TCB Interfaces

There are two types of interfaces to the TCB: gate interfaces, which present a subroutine interface to the calling process, and message interfaces, which are used by placing a message in a message segment and awaiting a response from the TCB process that performs the requested service.

Gates

This section lists the gates in the TCB by name, and identifies the number of entrypoints in each. Gates to the TCB, accessible by users:

Gate	# Entries	Gate	# Entries
dm_gate_	96	message_segment_	40
hcs_	198	rcp_	18
ioi_	12	restart_fault	2
mail_table_	3	shcs_	2
mailbox_	36	user_message_	2
mdc_	18		

Gates internal to the TCB, used only by ring one functions to invoke privileged ring zero functions:

Gate	# Entries
access_audit_gate_	25
admin_gate_	64
dm_hcs_	11

Gates internal to the TCB, used only by privileged processes to invoke privileged TCB functions:

Gate	# Entries	Gate	# Entries
dm_daemon_gate_	24	pnt_fs_gate_	7
hc_backup_	21	pnt_login_gate_	2

initializer_gate_	16	pnt_network_gate_	4
initializer_mdc_	25	rcp_sys_	20
mail_table_initializer_	2	user_message_admin_	2

Gates internal to the TCB, used by system administrators to invoke priviliged TCB functions:

Gate	# Entries	Gate	# Entries
dm_admin_gate_	29	queue_admin_	3
dm_hphcs_	2	r1_io_	18
hphcs_	137	r2_io_	18
installation_gate_	2	r3_io_	18
installation_tools_	18	r4_io_	18
mail_table_priv_	15	rcp_admin_	10
mdc_priv_	17	rcp_priv_	2
metering_gate_	7	system_privilege_	22
phcs_	25	tandd_	13
pnt_admin_gate_	12	user_message_priv_	5
pnt_priv_gate_	3		

Gates outside the TCB, used to invoke functions in non-TCB protected subsystems (see page B-8, "Ring Two Non-TCB Software"):

Gate	# Entries
forum_	49
forum_admin_	7
forum_chairman_	8
forum_notify_gate_	5
mail_system_	73

Other TCB Interfaces

The other interfaces to the TCB are message segment interfaces and event channel interfaces, rather than subroutine calls to gates. These TCB requests are made by formatting a message and placing it into a message segment request queue , and/or sending a interprocess communication wakeup over an event channel whose name is published in a globally readable data segment.

These are the Answering Service Request interfaoe, the Answering Service absentee request interface, the I/O Daemon output request interface, and the Volume Retrieval request interface. Additionally, there are several event channels used by individual processes to communicate with the Answering Service, to request

process termination, logout, reconnection, and system table installation. See page 24, "Dedicated TCB Processes" for details.

MPC Firmware

This is firmware for the microprogrammed controllers used to run disk, tape, and unit record I/O devices. It is effectively part of the hardware. The components listed below are those required for MR11.0 operation.

fw.dsc500.d500.r1
fw.dsc191.m191.t1
fw.mtp610.m610.r2
fw.mtp601.m601.r1
fw.mtc500.m500.v1
fw.msp800.msp8.a1
fw.urcmpc.ucrp.b2
fw.urcmpc.u400.l1
fw.urcmpc.ucmn.p2

FNP Software

Approximately 37,000 lines. This represents the code used to implement the communications protocols in the Front-end Network Processor (see page 8, "FNP - Front-End Network Processor").

mcs
mcs_autocall
mcs_bisync
mcs_g115
mcs_hasp
mcs_ibm3270
mcs_x25

Non-Evaluated Software

This section describes the three categories of software that were not included in the Multics evaluation.

Functions Excluded from the TCB

Approximately 44,000 lines. The following facilities present potential difficulties in security, or now-obsolete formerly trusted functions, and therefore were not evaluated. They must not be used in a B2 operating environment. They are the Carry faoility, Inter-Multics File Transfer, Cross Ring I/O, Heals, the Transaction Processing Monitor, and the Level-6 File Transfer

facility. The Multics ARPAnet support software, a special-order item not included in the standard system, is also excluded from this evaluation.

bound_carry_facility_
bound_cross_ring_io_
bound_ftp_ctl_
bound_heals_
bound_imft_
bound_16_ftf_
bound_tp_runtime_
bound_tp_tools_

Ring Two Non-TCB Software

Approximately 39,000 lines. Two standard products, Forum and the Mail System, operate in a ring-two protected environment that provides additional discretionary access control features. These are outside the TCB boundary, however, and were not evaluated.

bound_forum_mgr_
bound_mail_system_
bound_v2_forum_mgr_
forum_data_
forum_error_table_
forum_notifications_

Remainder of Multics Software

Approximately 1,638,000 lines. This represents the rest of the standard software in the Multics product: compilers, utilities, and user interfaces. This software was not covered by the evaluation.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT US Government & their contractors Other requests as determined by the NCSC.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) CSC-EPL-85/003	5. MONITORING ORGANIZATION REPORT NUMBER(S) S227,783
6a. NAME OF PERFORMING ORGANIZATION National Computer Security Center 6b. OFFICE SYMBOL (If applicable) C	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State and ZIP Code) 9800 Savage Rd. Ft. George G. Meade, MD 20755	7b. ADDRESS (City, State and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State and ZIP Code)	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT NO.
11. TITLE (Include Security Classification) Final Eval. Rep. HIS Multics MR11.0	
12. PERSONAL AUTHOR(S) Downs, Deborah, (the Aerospace Corp.); Wagner, Grant et al	
13a. TYPE OF REPORT final 13b. TIME COVERED FROM _____ TO _____	14. DATE OF REPORT (Yr., Mo., Day) 860601 15. PAGE COUNT 124
16. SUPPLEMENTARY NOTATION	
17. COSATI CODES FIELD / GROUP / SUB. GR.	18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) Multics MR11.0 Honeywell Information Systems NCSC EPL DoDCSC B2 Trusted Computer Sys. Crit.

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

The security features of Multics MR11.0 were evaluated against the requirements specified for a class B2 system in the Department of Defense Trusted Computer System Evaluation Criteria dated 15 Aug.'83. This report presents the findings of this evaluation.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL LTC Lloyd D. Gary, USA	22b. TELEPHONE NUMBER (Include Area Code) (301)859-4458 22c. OFFICE SYMBOL C/C12

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE