CSC-EPL-87/003

NATIONAL COMPUTER SECURITY CENTER

NATIONAL COMPUTER SECURITY CENTER

FINAL EVALUATION REPORT OF UNISYS CORPORATION A SERIES MCP/AS

RELEASE 3.7

5 August 1987

FINAL EVALUATION REPORT

UNISYS CORPORATION

A SERIES MCP/AS RELEASE 3.7

NATIONAL
COMPUTER SECURITY CENTER

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

August 5, 1987

CSC-EPL-87/003
Library No. S-228,515

CSC-EPL-87/003
Library No. S-228,515

FOREWORD

This publication, the Final Evaluation Report, UNISYS Corporation, A Series MCP/AS Release 3.7, is being issued by the National Computer Security Center under the authority of and in accordance with DoD Directive 5215.1, "Computer Security Evaluation Center." The purpose of this report is to document the results of the formal evaluation of UNISYS A Series MCP/AS Release 3.7 operating system. The requirements stated in this report are taken from Department of Defense Trusted Computer System Evaluation Criteria, dated December 1985.

Approved:

Eliot Sohmer

August 5, 1987

ELIOT SOHMER
Chief, Office of Product Evaluations
and Technical Guidelines
National Computer Security Center

ACKNOWLEDGMENTS

Team Members

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

Cornelius J. Haley

Jerzy W. Rub — National Computer Security Center, Fort George G. Meade, Maryland

Sheryl M. Luera

Mary D. Schanken — The Aerospace Corporation, Los Angeles, California

William B. Geer

Further Acknowledgments

Technical support was also provided by William A. Miller, W. Olin Sibert, W. Stan Wisseman, Paul R. Hager, and Dennis R. Hood.

CONTENTS

Page

EXECUTIVE SUMMARY

The National Computer Security Center (NCSC) has evaluated the security protection provided by the UNISYS A Series product line. The evaluated system is the A Series hardware (see page A-1, "Evaluated Hardware Components") running the Master Control Program/Advanced System Architecture (MCP/AS) Release 3.7 and the InfoGuard security enhancements (see page B-1, "Evaluated Software Components"). The InfoGuard security enhancements must be configured with the CLASS option set to S2, as described in the Security Administration Guide.(1) The security features of the A Series were evaluated against the requirements specified by the Department of Defense Trusted Computer System Evaluation Criteria [DOD 5200.28-STD], dated December 1985.

The NCSC has determined that the highest class at which the evaluated system satisfies all the specified requirements of the Criteria is class C2.

A system that has been rated as being a C division system provides for discretionary protection and, through the inclusion of audit capabilities, for accountability of subjects and the actions they initiate. Class C2 systems enforce a finely grained discretionary access control, making users individually accountable for their actions through login procedures, auditing of security-relevant events, and resource isolation.

The UNISYS A Series system consists of the MCP/AS operating system and InfoGuard security enhancements running on any member of the A Series product line. The system is capable of supporting from a few to a few hundred concurrent terminal users and their applications. MCP/AS is written in a high order language tailored to the ALGOL-based hardware architecture.

MCP/AS is a general purpose system that supports re-entrant/recursive multiprocessing, multiprogramming and virtual memory through its tagged memory and stack architecture. All processors in the A Series product line provide a single state for execution. Therefore, the system software on A Series is responsible for providing a self-protecting domain for the A Series Trusted Computing Base (TCB). Since the A Series TCB provides a self-protecting domain, the addition of a compiler, privileged program or other program specifically excluded in

---

(1) A Series Security Administration Guide, UNISYS Corporation, Document #1195195, July 1987.

appendix B beginning on page B-1, "Evaluated Software Components" is not an evaluated extension to the TCB. Such an extension would invalidate the C2 rating.

INTRODUCTION

In August 1986, the National Computer Security Center (NCSC) began a formal product evaluation of the UNISYS (formerly Burroughs) A Series product line running the Master Control Program/Advanced Systems architecture (MCP/AS) Release 3.7 and the InfoGuard security enhancements (see page B-1, "Evaluated Software Components"). The InfoGuard security enhancements must be configured with the CLASS option set to S2, as described in the Security Administration Guide. The goal of this evaluation was to rate the A Series system against the Department of Defense Trusted Computer System Evaluation Criteria [DOD 5200.28-STD] (the Criteria), dated December 1985, and to place it on the Evaluated Products List (EPL) with a final rating. This report documents the results of that evaluation. This evaluation applies to MCP/AS Release 3.7 and the InfoGuard security enhancements together. These products have been available from UNISYS since July 1987.

Material for this report was gathered by the evaluation team, through documentation, interaction with system developers, and experience using A Series 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 other sensitive information. In August 1985 the name of the organization 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 DoD Trusted Computer System Evaluation Criteria was written. 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 Conventions and Organization

Throughout this report acronyms and other terms traditionally capitalized by UNISYS will be capitalized. For clarity, certain UNISYS terms are underlined when first referenced in a group of paragraphs, to stress their importance.

This report consists of four major sections and three appendices. Section 1 is an introduction. Section 2 provides an overview of the system software and hardware architecture. Section 3 provides a mapping between the requirements specified in the Criteria and A Series features that fulfill those requirements. Section 4 presents the evaluation team's impressions of the system. Appendix A identifies the hardware models to which the evaluation applies. Appendix B identifies the software that is included and explicitly excluded from an evaluated configuration. Appendix C is a glossary.

SYSTEM OVERVIEW

This section introduces the subjects, objects and Trusted Computing Base (TCB) of A Series and the following section will describe A Series system software and hardware architecture.

Introduction

The name A Series refers to a hardware product line which currently includes the A1, A3, A4, A5, A6, A9, A10, A12 and A15. The operating system running on all A Series machines is Master Control Program (MCP). MCP with memory management enhancements is called Master Control Program/Advanced System (MCP/AS) and is designed for A Series machines. The specific models in the A Series product line are identified in Appendix A, page A-1, "Evaluated Hardware Components." MCP/AS does not run on predecessor systems (e.g., B6700, B6800, B7900). The primary difference between MCP and MCP/AS is the operating system's treatment of memory management. This evaluation applies only to the A Series product line when using MCP/AS with the InfoGuard security enhancements. Additionally, the InfoGuard security enhancements must be configured with the CLASS option set to S2, as described in the Security Administration Guide. The evaluated configuration is described in greater detail in the appendices on page B-1, "Evaluated Software Components", and on page A-1, "Evaluated Hardware Components." For the remainder of this report, the hardware and software just described will be referred to as A Series.

The interface between MCP/AS and the hardware within the A Series product line is consistent over its entire range. This range is defined in Appendix A, page A-1, "Evaluated Hardware Components." While there are differences in the internal architecture, these differences are not visible to a user. All processors in the A Series product line provide a single state for execution. This requires A Series system software to strictly control the creation of executable code. For this reason only UNISYS-supplied compilers may be loaded in the evaluated system. Programs compiled with user-invocable compilers have no access to the machine-specific security-relevant features.

Executable code can run on all A Series machines if compiled generically. Compiler options allow object code to be optimized to run on a specific model of machine. Object code is optimized by compilers in two ways. Some object code is optimized by using operators (i.e., machine instructions) available only on

particular models. (While all A Series models have the same operator set, predecessor systems do not support all of the Advance System architecture operators.) This type of optimization causes the compiler to mark the object code such that MCP/AS refuses to execute it on an inappropriate model. Another form of optimization takes into account the way a processor performs certain operations (i.e., a branch). Object code optimized in this way may be executed on any model; however, some models will perform more efficiently.

Trusted Computing Base

All models in the A Series product line have one hardware state. Therefore, A Series system software is responsible for building and maintaining a self-protecting domain of execution. The creation of the A Series self-protecting domain is accomplished through the strict control of executable code. Executable code can be created in only two ways: generated by compilers or restored from removable media (e.g., tapes). Access to A Series compilers, and therefore to the languages they implement, is controlled (see page 20, "Compilers"). Restoring an executable file from removable media is also controlled (see page 30, "Tape Handling Mechanism").

The A Series TCB is MCP/AS, system libraries, compilers, two message control systems and privileged programs. The InfoGuard security enhancements are implemented in a system library supported in the evaluated system. A system library is a support library that has been installed by an administrator for use by all users. Privileged programs are not subject to discretionary access controls. Programs obtain privileged status through an explicit action by the security administrator.

While it is typically not necessary to include compilers in a TCB, A Series compilers are an essential mechanism in the software TCB's self-protection. Three system languages - DCALGOL, DMALGOL and NEWP - provide a system programming capability, while user programming is accomplished with ALGOL, Fortran 77, Cobol 74, PASCAL and other languages. The compilers for these languages are specified beginning on page B-1, "Evaluated Software Components." The role of compilers in preserving the software domain is described on page 20, "Compilers."

Since the A Series TCB is responsible for building and maintaining a software domain, the addition of a compiler, privileged program or other program specifically excluded in the

list on page B-1, "Evaluated Software Components", is not an evaluated extension to the TCB. Such an addition would invalidate the C2 rating.

Hardware System Overview

The A Series systems are the current line of mainframe computers available from UNISYS. A basic configuration may consist of a processor, memory, an input/output module, and a maintenance subsystem. Additional processors, memory, and input/output modules can be added to the basic configuration increasing capacity and capability, and improving performance. Likewise, selected peripheral devices are available for customization of the system. These include disk drives, tape drives, printers, data communications equipment, and terminals.

The processor subsystem consists of a series of processors that operate together in executing system and user programs. While the processors in A Series all accept the same set of instructions, their translation mechanisms - implemented in microcode - differ. UNISYS does not allow its customers to modify such microcode.

The memory subsystem provides storage and handles all transfers of data between the main memory and the main processor. This subsystem consists of one or more memory control units and memory storage units, and the memory interface. The subsystem contains logic for detection and correction of memory errors. As seen by the user, the memory itself is laid out in 6-byte (48-bit) words. A system running MCP/AS has the capability to directly address 4 billion words of memory (24 billion bytes). The Actual Segment Descriptor (ASD) memory subsystem, implemented with MCP/AS, allows for such a large address space.

The ASD memory subsystem supports a single large memory structure that requires no partitioning and that is accessible to all processors in the system. All allocated memory areas are accessible and controlled through a central structure called ASD table. This table is a central repository of information for all memory in use by the system.

On some A Series systems, processor performance is also enhanced through the use of data and code cache memory. Cache memory is a limited-size, high-speed memory used for retaining the contents of recently referenced main memory locations. A memory design with A Series allows for a high hit ration within the cache memory improving the system performance.

The input/output subsystem manages all transfers of information between the operating system and peripheral devices. It consists of a series of specialized processors that are responsible for performing the actual transfers. I/O operations are initiated by the operating system but once started continue under the control of these specialized processors.

The peripheral devices are connected to the system and controlled through Data Link Processors (DLPs). Each DLP contains microcoded programs that accommodate the unique characteristics of the type of peripheral device it controls. A standard DLP exists for each type of peripheral device supported by A Series systems.

The maintenance subsystem acts as the interface through which the operator controls the hardware when the system is being initialized, configured, or stopped. This subsystem also provides a means to diagnose hardware problems.

Subjects

The subjects in A Series are tasks and jobs. A process in an A Series is commonly referred to as a task. Tasks are represented by a family of stacks (i.e., one or more stacks such as a parent and child stack) and a stream of executable code.

Batch processing in A Series is performed through jobs. A job is very similar to a task. An executing job is a task. Executing jobs are bounded by the same process protection and access controls as tasks. The difference between a job and a task is its method of creation. Jobs are subjects that are created by MCP/AS procedures from job queue entries (see page 15, "Job Control"). A task is a subject that is created by different MCP/AS procedures from MCS requests (see page 16, "Process Control").

The stacks of a task are TCB-generated and TCB-maintained data structures. These stacks represent the current addressing environment and the history of procedure entries that lead to the current environment. These stacks are a more complex data structure than the traditional first-in, last-out structure of a stack. Stacks and processes are described in more detail on page 36, "Stacks" and page 16, "Process Control."

Privilege

Users of A Series are identified by unique USERCODEs associated with individuals. Grouping of users is accomplished through the use of ACCESSCODEs. A user may specify an ACCESSCODE provided the user is an authorized member of the specified group.

A Series provides several forms of privilege. Privilege can be associated with a USERCODE or a program. The privilege associated with a USERCODE is used to define a distinction between privileged users (PU privilege), system users (SYSTEMUSER privilege), security administrators (SECADMIN privilege) and ordinary users. These privileges (i.e., PU, SYSTEMUSER, SECADMIN) are not hierarchical. Each privilege grants certain abilities to a USERCODE. User roles will be discussed in detail on page 26, "User Roles." The information associating a USERCODE with a type of privilege is stored in the USERDATAFILE. Tasks created on behalf of users possessing privilege assume the privilege specified in the USERDATAFILE. Tasks executing privileged programs assume privilege only while executing the privileged program. Task privileges are stored in the task's program information block (see page 16, "Process Control").

A privileged user (i.e., PU) is permitted to read, create or remove disk files stored under other USERCODEs, invoke certain operating system control interfaces (e.g., SETSTATUS), and create or alter USERCODEs (only if no USERCODE has SECADMIN privilege).

A system user has the privileges of an operator from any station. System users are not permitted the same capabilities as privileged users (i.e., PU) unless they also possess that privilege. For more detail on the duties of system operators see page 28, "System Operators."

The security administrator role is described in detail on page 26, "User Roles." Until the first Security Administrator is defined, a privileged user is allowed to define a Security Administrator. Once a Security Administrator has been defined and the privilege is authorized for the system, that privilege can only be granted to other USERCODEs by a Security Administrator. If the Security Administrator privilege is authorized for the system, then a Security Administrator USERCODE is required to execute security-critical functions, such as changing Security Options, changing the USERDATAFILE, or invoking the ODT commands and SETSTATUS calls that confer privilege or affect system security.

Programs may also be marked as privileged. The privilege associated with a program is the same as that associated with a privileged user (i.e., PU). Privileged programs must be marked as privileged by a security administrator. Any user may execute a privileged program provided that user has the appropriate access to the disk file containing the program.

Objects

The objects in A Series are disk files, tape volume families, printer files, card-reader files, communication files and databases. A Series provides a flat filesystem. Files are uniquely defined within the filesystem using a USERCODE, a file title and a FAMILY name. A USERCODE uniquely identifies a user to the system. A file title consists of between one and twelve names. All names except the last are referred to as directory names; and the collection of names forms the file name. Files are contained on a logical entity known as a FAMILY. A FAMILY is one or more disk packs. See page 11, "Disk Management", for more information concerning the structure of the filesystem.

While defaults exist, and are frequently used, for USERCODE and FAMILY name, all three identifiers are used in uniquely identifying a file. Files used by the system follow the same naming scheme, however they are associated with a USERCODE of "*" (star).

The term tape volume refers to one physical reel of tape. A tape volume family may consist of one or more tape volumes.

The term file is used by UNISYS to refer to two slightly different entities. A file is a collection of data on disk or tape that is part of the filesystem. A FILE refers to the logical object accessed from within programs. A FILE can be thought of as a conduit.

Printer FILES are used by programs to write to a printer. However, user programs are not permitted to write directly to the printer without operator intervention. Instead, data is written through a printer FILE into a temporary location known as a printer file. A printer file is spooled by the system to disk (or tape). By system option, these files are spooled either to the user's own disk directory or to a non-USERCODEd directory. In the evaluated configuration, spooled files inherit the USERCODE of the user or task under which they are created and are protected as private by default.

Card-readers are used in two ways: for data entry and for job entry. When used for data entry a card-reader FILE is used by a program to read from card readers.

In the evaluated configuration, all card readers are secured. This requires a USERCODE and password for all jobs submitted from a card reader. Cards are also read as job decks by the Work Flow Language (WFL) compiler. Data decks within a job deck may be incorporated into the WFL jobfile or transcribed to disk files. Data within jobfiles represent spooled card-reader files. Spooled card-reader files are accessible only to tasks initiated within that job. Data transcribed to disk create new private disk files under the USERCODE of the task.

Three different types of communication FILEs are possible in A Series computers. The FILE types are: REMOTE, Operator Display Terminal (ODT), and Port.

REMOTE FILEs are connections between a program and a station (i.e., terminal). The controlling Message Control System (MCS, see page 21, "Message Control System") of a station determines if a REMOTE FILE may be opened at a station. The MCS will only open a REMOTE FILE under one of the following two conditions:

- the station has already been assigned to the USERCODE requesting the REMOTE FILE;
- the station has an authenticated user that accepts the connection.

ODT FILEs are similar to REMOTE FILEs except that they are connections to the operator console. An ODT FILE may only be opened with consent of the operator.

Port FILEs are used for inter-process communications. A Port FILE contains one or more subports. Each subport allows one conversation. File attributes, including USERCODEs, are used to match and interconnect these ports. These ports and the associated subports are protected by MCP/AS. Port FILEs may only be connected if each FILE is declared with identical names, and each file specifies both USERCODEs involved in the connection.

A Series supports a data management mechanism called DMSII. A DMSII database consists of several disk files: a control file, a number of data files, and an executable codefile known as the access control routines (ACR). The mechanism for providing disk file discretionary access controls is used to provide access control to the data files, control files, and ACR routines of a DMSII database. Separate guardfiles are attached to each of the

files associated with the database. Read permission to the control file and execute permission to the access control routines is necessary for users desiring to access the database through the access control routines. To ensure that users only access the data files using the DMSII database, a guardfile must be attached to the data files to deny direct read or write access. The discretionary protection provided on the three database files is capable of requiring users to access database information through the access control routines.

System Software Overview

A Series system software consists of MCP/AS, support libraries, compilers, and two message control systems (MCS). This section describes A Series system software followed by several security-related topics.

Master Control Program

Master Control Program/Advanced System (MCP/AS) is written in a privileged language called NEWP. MCP/AS is responsible for disk and memory management, hardware interrupt processing, job and process control, I/O operations and other system overhead functions.

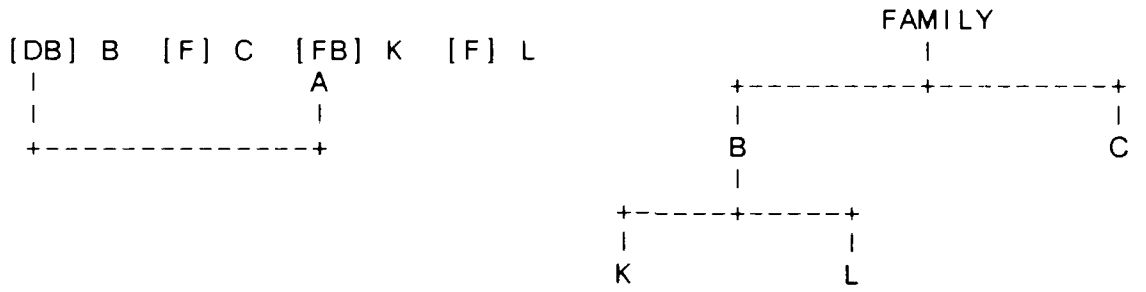
Disk Management

Areas on disk are categorized as in-use or available. In-use areas are arranged into the filesystem. Addresses of available areas are collected in tables for future allocation as part of the filesystem.

One or more disk packs may be organized into a logical entity known as a FAMILY. Each FAMILY has associated with it an MCP/AS-managed table known as the FLAT directory (FLAT). Entries within the FLAT are disk file *headers*. A *header* contains information describing the attributes, location and name of a file. File attributes in a *header* include *header* size, opencount, filekind, security information, block size, record size, units, timestamps, row size and the end of file pointer. The *header* also contains row address words which contain the address of file areas on the disk. The row size and row address attributes in the *header* provide a base and bound that defines the areas comprising a file.

One FLAT contains a *header* for the system-wide ACCESS STRUCTURE. The ACCESS STRUCTURE is used to quickly locate a file's *header* within a FLAT. The ACCESS STRUCTURE has two parts: the Pack Access Structure (PAST) and the File Access Structure (FAST). The PAST contains pointers into the FAST for each FAMILY known on the system. The FAST contains a logical tree-structured representation of the files and directories on each FAMILY.

Within the FAST the concept of a brother is used. Brothers are entries in a FAMILY at the same level in a tree having the same predecessors. Brothers are stored contiguously in the FAST. Both directory and file entries may signify that a brother exists. Directory entries in the FAST point within the FAST to the first child of that directory. File entries contain a pointer to the header of a file in a FLAT. Figure 1 is an example of the contents of a FAST for the FAMILY shown to the right.



In the above figure a "D" denotes a directory, an "F" denotes a file, and a "B" signifies a brother exists. The arrow indicates the offset in characters to the next level (or first child).

Figure 1.

The security information stored in a header includes four file attributes: SECURITYTYPE, SECURITYUSE, SECURITYGUARD and SENSITIVEDATA. File attributes are assigned, listed or modified through Message Control Systems or by using a programming language (e.g., WFL, ALGOL, etc.). In addition, three security file attributes are used exclusively for volume families. The volume attributes used to secure volume families are described beginning on page 30, "Tape Handling Mechanism."

The SECURITYTYPE file attribute specifies who may access a file. Files are considered either public, private, guarded or controlled. Public files are accessible (i.e., as defined by the SECURITYUSE attribute) by all users of the system. Private files are accessible only by their owner (i.e., creator). If the SECURITYTYPE is not specified when a file is created, the default value is private. By specifying a file as guarded or controlled, a file owner has chosen to selectively assign access rights to that file. These access rights are determined from the access information contained in a guardfile. The owner of a file that is guarded is allowed access regardless of the access information in the guardfile. The access for all other users is determined

by the guardfile. When a file is controlled, the guardfile is examined prior to granting access to any user. Guardfiles are explained in more detail on page 29, "Guardfile Mechanism."

The SECURITYUSE file attribute is examined only on public files and specifies the way an unprivileged user or unprivileged program can access a file. The SECURITYUSE attribute can take on a value of read-only (IN), read-write (IO), write-only (OUT) or execute-only (SECURED). The default for the SECURITYUSE file attribute is read-write.

The SECURITYGUARD file attribute identifies the name of the guardfile when the file is guarded or controlled. If a guardfile is not correctly identified, a guarded file is protected as private, and no access is allowed to a controlled file.

The SENSITIVEDATA file attribute causes the disk area assigned to the file to be overwritten with an arbitrary pattern before the space is returned to the system for reallocation. The SENSITIVEDATA attribute can be manipulated by a user. (In an evaluated configuration disk areas are always overwritten upon allocation to a user.) This attribute allows a user to explicitly overwrite a file upon deallocation.

Another of the attributes in a file's header is filekind, which can take on values in five different categories: system, compiler, codefile, program symbol, or data. Files with filekind system contain data files and object code belonging to the operating system. A file with a filekind of compiler is a compiler. Files with filekind compiler or codefile contain object code. A compiler is always executable. A codefile while it contains object code may not be executable. A codefile may be non-executable if the source that it was created from contained certain unsafe constructs (see page 20, "Compilers"). The tag field of every memory word, described on page 35, "Tag Architecture", is seldom stored on disk. A tag field is stored on disk only if it has been forced out of memory by an overlay operation. When files are read into memory from disk, MCP/AS recognizes the file's filekind and assigns appropriate tags during the read operation.

Only MCP/AS can mark a file as having a filekind of system or compiler. Compilers mark the object code files they produce as having a filekind of codefile. User data and user program source are stored in files having a filekind of data or program symbol respectively. Users are allowed to read and write files with filekinds of program symbol or data.

Memory Management

A Series aggregates contiguous words of memory into segments. There are two views of segments in A Series, virtual and actual. Virtual segments are defined by a Code Segment Descriptor (CSD) or Data Descriptor (DD). An actual segment is a contiguous group of words, and is defined by an Actual Segment Descriptor (ASD), to which a CSD or DD points.

A virtual segment is the memory-word grouping visible to most machine instructions. A virtual segment defined by a CSD contains object code while a virtual segment defined by a DD provides space for arrays, records and other areas that occur in programs.

Actual memory segments in A Series are further categorized as available, save or overlay. Available segments are segments that are not currently being used. Save segments are in-use and may not be moved or copied to disk. Overlay segments are also in-use but may be either moved within memory or copied to disk.

Actual segments may vary in size (up to a maximum size of 1 megaword: 6 Megabytes) but every segment is described by an ASD. An ASD is a three-word structure containing the absolute memory address of the first word of the segment, status flags and the length of the segment. MCP/AS maintains actual segment descriptors in the ASD table. The base of this table is pointed to by a register, the ASD_table_base register.

An ASD entry is composed of three words. The first word (ASD1) identifies a segment as either stack or non-stack, and either in memory or absent. It also specifies an address for the actual segment. If the segment is absent the address field of ASD1 contains a value identifying the segment as save or overlay. The second word (ASD2) specifies the actual segment length in words. The third word (ASD3) references the original data descriptor that "owns" the ASD.

MCP/AS links available memory segments into two linked lists. One list identifies all segments that can be used as save memory and the other list identifies segments that can be used as overlay memory. When a program attempts to address code or data that is currently on disk, a hardware interrupt is generated. MCP/AS services the interrupt as described on page 15, "Hardware Interrupt Processing." While servicing the interrupt, MCP/AS traverses the appropriate available segment list attempting to find a segment large enough for the information on the disk. If

an appropriate segment is found the segment is removed from the available list and overwritten. Attempts to address data that are uninitialized (i.e., previously unused variables) follow the same procedures except the memory segment is first cleared. The memory is cleared by writing a zero into each memory word and marking the word as uninitialized data (see page 35, "Tag Architecture"). This provides hardware-supported protection against the accidental or malicious use of uninitialized data. If an available segment is not found, a demand overlay is performed. Demand overlay creates a memory segment of sufficient size by moving overlay segments either within memory if possible, or to disk if necessary.

Hardware Interrupt Processing

Hardware interrupts are processed by calling an interrupt handler within MCP/AS. The interrupt handler determines which hardware interrupt has occurred and calls the corresponding MCP/AS procedure to take specific action. Examples of hardware interrupts are non-present data, stack overflow, processor-to-processor communication, I/O complete and invalid reference.

MCP/AS, being capable of executing on the entire A Series hardware product line, requires a different interrupt handler for different models. During initialization, the system places a pointer to the interrupt handler three words from the base of the MCP/AS stack (i.e., offset 3 relative to D[0], see page 36, "Stacks").

Job Control

Batch processing on A Series is performed through jobs. A job is a program compiled by the WFL compiler. Jobs control the sequence of actions performed by tasks. A job may enter the system via a card reader, an operator console, a Message Control System, or be spawned by an existing task. When a job is compiled by the WFL compiler, the syntax of the job control statements is checked. The compiler then translates the job control statements into machine code. This machine code is placed into a jobfile which also contains space for logging, restart information, copies of control cards, and data decks. The MCP/AS procedure for auditing, WRITELOG, makes entries for non-security related entries into both the jobfile and the system log. Security-related log records are written to the system log only. After a job is compiled, it is entered into a job queue. Jobs entered into the job queue are started by the MCP/AS and

cause the creation of independent authenticated tasks (i.e., processes). When a job ends, a backup file may be printed that contains the job related log entries and control statements.

Process Control

A process is a program in execution; its stack can be considered the instantaneous state of that process. As stated earlier UNISYS refers to a process as a task; however, this section will use the term process. This section describes the life of a process from creation to termination.

First, a program information block (PIB) is allocated and filled by MCP/AS. Attributes of the new PIB are assigned from the PIB of its parent, the codefile and occasionally the job. MCP/AS assigns a stack number and a mix number to the PIB. This mix number is subsequently used by the auditing mechanism to identify the process responsible for an audit record.

A memory area is obtained for use as a process stack. While building the stack, MCP/AS places calls to NORMALBOJ and NORMALEOJ into the stack. The call to the MCP/AS procedure NORMALEOJ is placed into the stack such that when this process' code has completed, NORMALEOJ is invoked. The remainder of the stack is built by MCP/AS such that when the stack becomes active its first action is to call the MCP/AS procedure NORMALBOJ. MCP/AS then places this new process into the ready queue for the first time.

Once in the ready queue, the new process waits to begin execution through a normal process switch (see page 39, "Process Switching"). After a process switch has occurred, NORMALBOJ is executing on the new stack. NORMALBOJ finishes building the stack, generates an audit message signifying the creation of a new process and exits. When NORMALBOJ exits, the new process begins execution of its code segment.

After the process' code has completed, an exit into the MCP/AS procedure NORMALEOJ occurs. NORMALEOJ generates an audit message signifying the termination of the process, thereby disassociating the mix number and the process. NORMALEOJ releases as much of the stack's memory and disk space as possible, but since it is currently running on the stack not everything can be released. A call to a system process is made requesting the release of the current stack, and NORMALEOJ exits. After NORMALEOJ exits, the system process releases the remaining stack and PIB space.

Logical I/O

When a program is compiled, information about files the program intends to access is encoded into file information blocks (FIBs). The FIB contains the file's name internal to the program and a representation of the file's declared and default attributes. Disk file attributes include familyname, filekind, filetype, fileuse, newfile, protection, SECURITYGUARD, SECURITYTYPE, SECURITYUSE and SENSITIVEDATA. A FIB contains a pointer to a label equation block (LEB). A LEB contains a representation of a file's attributes that are modifiable at run-time.

When a program is executed, its stack building code generates a pointer in the stack to the FIB. References to the file are made using the FIB. Before a file can be accessed, it must be associated with a physical file (i.e., opened). An MCP/AS procedure, FIBOPEN, associates the logical file defined by the FIB with a physical file in the filesystem after performing an access check. This procedure also generates an audit record, places an I/O control block into the FIB, and sets up the buffers to handle the I/O.

Physical I/O

Users on A Series systems are not permitted to directly access the physical I/O subsystem. All user I/O actions are performed by MCP/AS procedure calls generated by the compilers. I/O requests are passed to the Logical I/O module, from which they are passed to the appropriate Physical I/O routine. This routine creates or modifies an existing data structure (Input/Output Control Block, IOCB), which contains all the information required to service the I/O request (including unit, address, and input/output buffer).

The IOCB is then passed to the I/O processor: either MLIP (Message Level Interface Processor) or HDU (Host Dependent Unit). The I/O processor is responsible for queuing the I/O for the appropriate unit queue. An HDU on an A15 or A12 is responsible for determining the destination DLP (Data Link Processor) in the case of units with multiple paths. On other A Series machines, path selection is performed either by the MLIP or by the Physical I/O module. Both I/O processors pass the I/O request to the appropriate DLP, which is responsible for controlling the actual data transfer.

Upon completion of the I/O, the IOCB is placed in a result queue for further processing. The Physical I/O module determines the failure or success of the I/O, performing any retries as necessary. To signify completion of the I/O, an event is caused to provide notification to the I/O requester.

Remote terminal support can be done in a numbers of ways. The three which are possible with UNISYS A Series are Multidrop Lines, Burroughs Network Architecture (BNA), and Direct Lines. Multidrop Lines which allow one physical line to address several terminals by multiplexing (either in time or frequency domain) on the same line cannot uniquely identify each terminal. This inability to correctly identify each terminal makes it impossible to meet requirements for identification and authentication and auditing. Therefore Multidrop Lines are specifically excluded from use in the evaluated system. The BNA software was not evaluated (see appendix B, page B-1, "Evaluated Software Components"). Direct lines connect each terminal to a separate individual port of the system. Direct lines can be uniquely identified and are therefore the only acceptable method of attaching terminal to the A Series

Support Libraries

The Library mechanism allows the creation of procedures that are available by runtime linkage to tasks (i.e., processes) in the system. This mechanism allows users to share the same executable code and addressing environment while maintaining separate stacks. Depending on its functionality, a Library may allow its callers to share data. Such a capability would have to be explicitly implemented within a Library. The Library mechanism is available to ordinary users and as a means of establishing system-wide functionality. Users may create Libraries for their own use. A Library may also be installed by an administrator as a System Library. All users may use a System Library.

A Library runs as a separate task. Each Library has its own stack, task attributes and addressing environment. The global variables of a Library are indirectly visible to a client (i.e., calling program). Library procedures run in the stack space of the client.

When Library procedures are called, a dummy stack frame is built on top of the client stack. This dummy stack frame represents how the client stack has declared the Library procedure. If the declaration in the client stack matches the actual declaration of the Library procedure, then the dummy stack frame is linked to the executable code of the Library procedure. The Library

procedure then runs on the stack of the client. That is, the Library procedure behaves as if it had been part of the client's program.

The linking of a Library to a client is performed at runtime. When a program first calls a Library procedure, all procedures within the Library that the program may use are checked. Programs must declare the Libraries and procedures to be used including all parameters to the procedures. MCP/AS compares the declarations of Library procedures made by the client with the actual Library procedures. If an inconsistency is identified, the linking fails and the client is terminated.

A Library running on the stack of the client has no visibility of the global variables of the client. A procedure's variable references are resolved at compile time when the Library has no way of knowing the global environment in which it will run. Similarly, a client has no access to the variables of the Library because all variable references are resolved at compile time. A Library, however, can access task attributes of the client and variables passed as parameters. References made to task attributes from within a Library procedure are considered references to the task attributes of the client. Only the initiation and termination code of a Library can access the task attributes of a Library.

Libraries do not behave like typical executable programs. MCP/AS initiates the Library stack after the first potential client process makes a call on the Library, and terminates the Library stack if no client is using the Library. The stack of a permanent Library, however, remains in the system even if there is no active client process linked to the Library.

The first time a procedure in a Library is called, the client is suspended. MCP/AS initiates a stack for the Library and begins execution of the Library's initialization code. The Library then declares the procedures it will export to its clients and performs a freeze. The freeze stops execution of the Library and causes the client to resume. When the client resumes, a procedure entry to the Library procedure has been completed. The stack frame for the Library is on top of the client stack with the addressing environment of the Library rather than the addressing environment of the original process.

Libraries may be called by more than one client at a time. Should a call be made on a Library that has already executed a freeze, execution continues with the Library executing on the new

client's stack. For this reason, the overhead for initiating a stack and running a Library's initialization code is only incurred once if a Library is permanent.

Compilers

Compilers in A Series are expected to perform the same function as compilers on any other system. Compilers are expected to accurately implement the constructs of their language. It is the constructs within these languages that provide capabilities to users. The languages supported on A Series are categorized as privileged and unprivileged. The semantics in certain languages provide the user with an ability to manipulate data at varying levels of abstraction. For instance, a user capable of compiling and installing a program written in NEWP, a privileged language, has the ability to directly manipulate a 52-bit word (i.e., a 4-bit tag and 48 bits of data). A user with access to a compiler for the ALGOL language (an unprivileged language) may directly operate on only the 48 data bits of a word. Such programs can and do sometimes also manipulate the 4 tag bits, but only with compiler-determined code sequences that precisely support the high-level language semantics. For a detailed description of tags see page 35, "Tag Architecture."

Three privileged languages - DCALGOL, DMALGOL and NEWP - can be used to develop system capabilities. Privileged languages contain constructs that allow access to interfaces and data types not available to ordinary languages. Only NEWP contains a construct called unsafe. This construct is used to define blocks of code that could be used to compromise system integrity or bypass security policy. An unsafe block allows additional data types (e.g., DESCRIPTOR and WORD). NEWP also provides access, within unsafe blocks, to address equation, NULL procedures, SAVE arrays, special uses of segment identifiers and other intrinsics (i.e., built-in function of a programming language). Any use of unsafe constructs causes the resulting code file to be marked as not executable until installed by the security administrator (using the XP, executable program, command). Access to privileged languages is limited by controlling access to their compilers through application of discretionary access controls. Access to these languages (and compilers) must be restricted to trusted users, since the languages are capable of creating code that could subvert system security. The Security Administration Guide (SAG) states that these languages (and compilers) should be restricted to trusted individuals. The Security Administration Guide is A Series Trusted Facility Manual (TFM).

Ordinary users produce executable code by using one of the system's unprivileged-language compilers (e.g., ALGOL, PASCAL, Cobol 74, Fortran 77). These unprivileged-language compilers do not produce unsafe constructs. This is done by allowing the user direct access to only 48 bits of information within a word (the user cannot access tags), separating the user's code and data, and providing access only to MCP/AS interfaces intended for use by ordinary users.

While compilers are not responsible for making access checks on files, compilers are responsible for ensuring that all accesses to a file are through valid descriptors. A valid descriptor is created by the MCP/AS procedure FIBOPEN. Compilers are responsible for causing a file to be opened by MCP/AS before it may be accessed. Therefore, MCP/AS makes the access checks at run-time as a file is being opened.

Message Control System

A message control system is a program that interfaces directly to the data communications subsystem. The A Series TCB contains two MCSs: the Communication Management System (COMS) and the Command And Edit Language (CANDE). An MCS is considered part of the A Series TCB because it is responsible for authenticating users, separating user I/O and generating audit messages. In addition, an MCS is considered a *privileged process* able to access any file, execute system queries, and execute system control interfaces (e.g., SETSTATUS, SYSTEMSTATUS). In general, MCSs are written in DCALGOL.

Sites may develop their own MCSs. However, such an MCS could compromise system security. The use of an MCS not specified in appendix B (see page B-1, "Evaluated Software Components") would constitute an extension to the TCB. As pointed out in the UNISYS Security Administration Guide, extensions to the TCB invalidate the C2 rating. MCSs cannot be created by normal users since they must be written in a privileged language. MCSs must also be installed at a site by a system administrator before they can be run by users. Installation at a site entails naming an MCS in the datacomm network definition (i.e., station configuration) for a system.

The following sections describe the MCSs in A Series TCB in terms of functionality, structure and user separation.

Communication Management System

The COMS message control system provides a window feature that allows users to operate multiple program environments from one station. One station may have multiple windows (e.g., a MARC - Menu Assisted Resource Control - window and a CANDE - Command AND Edit - window). Windows are of three types: MCS, remote or direct. An MCS window links the station to an MCS (such as CANDE). A remote window links the station (via files) to a specified program. A direct window links the station to transaction processing programs under the control of COMS. COMS automatically defines direct windows for MARC and UTILITY and an MCS window for CANDE.

A COMS window dialog is the connection between the station and the program that is currently communicating with the station. Each window may have up to eight active window dialogs.

Menu Assisted Resource Control (MARC) is a specialized transaction processing program designed for use by either an ordinary user or system operator. MARC serves as the user interface for COMS. User authentication is performed by MARC if COMS is defined as the station's default MCS. Every station connected to A Series has a default MCS specified in the system's network definition.

Once authenticated by MARC, the COMS message control system passes the identity of a user to other COMS windows and window dialogs. COMS windows and window dialogs may be created with a USERCODE (e.g., login identity) different from the USERCODE initially used to log-on under MARC. The creation of a window dialog under another USERCODE requires authentication of the new USERCODE (i.e., a valid USERCODE/PASSWORD). Subsequent windows or window dialogs revert to the original USERCODE. Users may log-off of either individual windows or window dialogs by using the BYE command from the window or window dialog. A user may log-off the system by executing the BYE command from the initial MARC window.

The COMS UTILITY window allows the security administrator, or a user designated by the security administrator, to define windows and specify the number of window dialogs, for a any window, that may be attached to each station. Each window dialog is viewed by the system as a unique station. Use of the COMS UTILITY window is limited to users and stations defined as being control capable (i.e., users or stations with no restrictions on the use of the COMS commands). In the evaluated configuration, UTILITY window

access is restricted to users designated as COMSCONTROL in the USERDATAFILE. The SAG states that access to the UTILITY window be limited to trusted individuals who require the use of that window.

An MCS window connects a station to either COMS, CANDE, or other user-written message control systems. In the evaluated configuration a COMS MCS window will connect a station to either CANDE or COMS only.

Within COMS there is the ability to restrict USERCODEs to particular stations, windows or trancodes (transaction types defined by an application). Similarly, stations may be restricted to particular windows or trancodes.

COMS utilizes the Library mechanism to maintain separation of users, and integrity of the MCS is assured in the same manner as support Libraries.

Command And Edit

The Command And Edit (CANDE) MCS provides users with an editor and a command level interface to MCP/AS. Unlike many other editors, only one instance of CANDE is executing on the system (i.e., not one per process). Therefore, CANDE interfaces directly with the data communication system on behalf of many users, and is responsible for separation of user data. CANDE provides a user with a command interface to MCP/AS that allows users to save files, attach guardfiles, execute programs and obtain status information along with other miscellaneous functions.

The CANDE MCS is structured such that CANDE's most global program block is a single stack that may PROCESS off dependent stacks. The single global stack is responsible for some simple control commands, datacomm functions, error recovery and distributing work to CANDE's dependent stacks. CANDE's dependent stacks perform editing, file creation and file deletion functions. Some CANDE commands cause new tasks to be initiated. These tasks are independent stacks performing actions on behalf of some user.

Another of CANDE's responsibilities is the authentication of users. CANDE requires that a user provide a USERCODE and PASSWORD before being granted access to other CANDE commands. During the process of authenticating a user CANDE is responsible for the generation of audit messages reflecting either a successful or unsuccessful login attempt.

USERDATA

USERDATA is the MCP/AS procedure which provides the system interface to user identification data maintained by the TCB in the SYSTEM/USERDATAFILE. USERDATA is an intrinsic function in ALGOL and its derivatives DCALGOL and DMALGOL.

The ALGOL USERDATA function calls USERDATA to perform any of the following fifteen functions. USERDATA returns a boolean value for all functions.

	Function and Call Value	Who May Access
1.	Examine User - Provided Entry	ALGOL Users
2.	Fetch and Examine Own Entry	ALGOL Users
3.	Validate Own USERCODE/PASSWORD	ALGOL Users
4.	Validate Own USERCODE/CHARGECODE	ALGOL Users
5.	Reserved for Future Implementation	
6.	Change Own Password	any user
7.	Modify/Create/Delete/See Entry	SECADMIN
8.	Create Entry using Model	SECADMIN
9.	Privileged Fetch and Examine	MCS or MCP/AS
10.	Reserved for Compiler Use	Compiler
11.	Validate Own ACCESSCODE/PASSWORD	ALGOL Users
12.	Change Own ACCESSCODE/PASSWORD	ALGOL Users
13.	Remoteuser (0) Return Local - Alias USERCODE (1) Rebuild Remote User Entry	BNA
14.	Locate *SYSTEM/USERDATAFILE	any user
15.	Reserved for Internal Use by MCP/AS	MCP/AS

USERDATA is called in response to the HELLO command by both MARC and CANDE in order to identify and authenticate users before granting access to system resources. The ODT command MU invokes the DCALGOL intrinsic MAKEUSER, which calls USERDATA to define a new user of the system. The USERDATAFILE is created by the MAKEUSER utility. MAKEUSER maintains the USERDATAFILE by way of USERDATA.

SECADMIN is a privilege that can be granted to a USERCODE via MAKEUSER. Until the first Security Administrator is defined, a privileged user is allowed to run MAKEUSER to define a Security Administrator. Once a Security Administrator has been defined and the privilege is authorized for the system, that privilege can only be granted to other USERCODEs by a Security Administrator. If the Security Administrator privilege is authorized for the system, then a Security Administrator USERCODE is required to execute security-critical functions, such as changing Security Options, changing the USERDATAFILE, or invoking the ODT commands and SETSTATUS calls that confer privilege or affect system security.

USERDATAFILE

The user data resides in a file named *SYSTEM/USERDATAFILE (USERDATAFILE), which contains one entry for each valid USERCODE. All access to the USERDATAFILE is provided by a set of MCP/AS procedures that are accessible as ALGOL or DCALGOL intrinsics. One of these, the USERDATA intrinsic, reads or updates the USERDATAFILE as necessary. Some of the functions of USERDATA are not provided to ordinary users. It is the responsibility of USERDATA to determine the privilege of the caller, and provide the appropriate access.

The MAKEUSER utility is the security administrator's principal tool for manipulating the USERDATAFILE. This program can create a new USERDATAFILE, copy or reinstate an old one, or examine and modify the current file by using the USERDATA intrinsic. Appropriate safeguards and interlocks are provided to permit file maintenance while the system is in normal operation.

Since the current USERDATAFILE provides the basis for protecting access to system resources, it is necessary that the file itself be protected from unwarranted change and accidental or deliberate removal, and that the file be backed up to protect against loss. Those aspects of the USERDATAFILE that could be used to violate system security can be changed only by the security administrator. The *SYSTEM/USERDATAFILE is marked as a SYSTEMFILE and it is kept open by MCP/AS. A SYSTEMFILE cannot be changed or removed. The only way for the USERDATAFILE to have its SYSTEMFILE designation changed is through the DCALGOL USERDATAFREEZER intrinsic. USERDATAFREEZER is called by MAKEUSER and causes MCP/AS to close the file and turn off the SYSTEMFILE designation. This allows a backup copy of the USERDATAFILE to be installed. Only the SECADMIN can invoke USERDATAFREEZER.

The USERDATAFILE identifies valid USERCODEs, PASSWORDs, ACCESSCODEs, APASSWORDs, CHARGECODEs and other user information. Each USERCODE in the USERDATAFILE has associated with it a SHOWFILE attribute, that can only be set by the SECADMIN. This attribute determines whether an ordinary user has the capability to display the names of public files belonging to another USERCODE. The USERCODE and ACCESSCODE passwords stored in the USERDATAFILE are encrypted. The USERDATAFILE used by the system must have a file title of *SYSTEM/USERDATAFILE.

The USERDATAFILE may only be altered by those USERCODEs with the SECADMIN option set (however, any user may change his own password). In FIBOPEN, the ACCESSRESTRICTION bit in the USERDATAFILE file header is checked to prevent direct access to the USERDATAFILE by an unauthorized user. All attempts to modify the USERDATAFILE are logged by MCP/AS in the system SUMLOG file. The logged information contains the specific changes made to the USERDATAFILE.

User Roles

A Series is designed to support several different types of users, each providing a different level of user functionality. Listed below is a description of six user types and their corresponding roles. Superusers, * (asterisk), USERCODEs are not discussed since they are not allowed in the evaluated configuration.

Security Administrator

The SECADMIN option must be authorized for the system in the evaluated configuration. Individual USERCODEs must then have another option (also called SECADMIN) enabled to become security administrators. The security administrator (also SECADMIN but for ease of reading further references will be to the security administrator) can modify and query the SYSTEM/USERDATAFILE, and may execute several other security-related commands. The security administrator has the following capabilities:

1. The ability to modify and interrogate the data base that defines users and their corresponding access privileges (i.e., the SYSTEM/USERDATAFILE). This includes use of the MAKEUSER utility, which is used to create, delete and modify USERCODEs stored in the USERDATAFILE.

2. The ability to issue the command, ??SECAD- (??SECAD- is a command). This is an ODT primitive command, which removes the system SECADMIN option.

3. The ability to invoke SETSTATUS calls which confer privilege or affect system security.

4. The ability to use the MARC DIRECTIVE command. The DIRECTIVE command allows execution of administrator defined, and created procedures in addition to the standard procedures provided by MARC.

5. The ability to mark a non-executable codefile as executable. This function, XP (Executable Program), is used to make unsafe programs or codefiles restored from tape executable.

6. The remaining commands available to a security administrator are:

   - CF (Configuration File)
   - DL (Disk Location) with the LOG or USERDATA option
   - HU (Host USERCODE)
   - ID (Initialize Datacomm)
   - LG (Log for Mix Number)
   - LOGGING (Logging Options)
   - MC (Make Compiler)
   - PP (Privileged Program)
   - REMOTESPO (Activate REMOTESPO) with the OK option
   - RESTRICT (Restrict file, unit, volume, host)
   - SECOPT (Security Options) which allows the setting of a number of system security features, such as scrubbing data on disk files before reuse, ownership of printer backup files, and automatic generation of passwords
   - SL (Support Library)
   - SR (Secure Reader)

Privileged Users

Privileged users are trusted individuals using USERCODEs that are designated as privileged. A privileged USERCODE may unconditionally create, read, write, execute and destroy other user's files. A privileged USERCODE may also access MCP/AS

procedures (e.g., GETSTATUS, SETSTATUS, and DCKEYIN) not available to ordinary users. Access rights of privileged USERCODEs are restricted when the system-wide SECADMIN option is set and at least one USERCODE is designated as a security administrator. The SAG states that privileged status should only be given to a USERCODE when it is necessary to grant this high level of system access, and when the user can be trusted.

System Operators

System operator duties include communicating with the Controller and Master Control Program/Advanced System (MCP/AS), initializing the datacomm subsystem, taking system dumps, monitoring system information, starting utilities, handling printouts, and performing similar tasks. System operators communicate with the system through ODTs (operator display terminals). It is possible to have more than one ODT in the evaluated configuration. These ODTs must be configured as restricted, thereby requiring login under COMS/MARC control. Restricting ODTs forces all entries from an ODT to be made through MARC. At login time the USERDATAFILE is checked to see if the USERCODE is a system user. A system user has the privileges of an operator from any station.

The SAG contains a chapter on controlling access to the physical system, as well as policies that operators should follow. Also the ODT Manual(1) describes all security-relevant operator commands.

System Users

A system user may perform functions normally associated with the system operators; i.e., remote ODT requests via MARC from the USERCODE are given the same privileges as ODT requests from an operator at the ODT. However, the scope of such functions is determined by the system user's possession of privilege.

---

(1) A Series Operator Display Terminal System Commands Reference Manual, UNISYS Corporation, Document #1169612, June 1987.

Security Message User

Any user may be designated by the security administrator as a security message user. A security message user - while logged in to a station designated as a security station - receives all system security violation messages. Users performing auditing functions may also be authorized as the security message users.

Ordinary Users

Ordinary users include application programmers and end users that access A Series. These users have no special privileges and are prohibited from gaining access to privileged constructs. All users (including privileged users, system users and security administrators) are uniquely identified in the USERDATAFILE. Ordinary users are responsible for maintaining secrecy of their passwords, protecting their files and properly allowing other users to access their files.

Guardfile Mechanism

A Series implements an access control list called a guardfile to specify discretionary access control to an object. Guardfiles can be attached to disk files, volume families and DMSII databases. A guardfile is like any other file, its owner can define it as being public, private guarded or controlled. Guardfiles can be shared with other users by specifying the title of the shared guardfile in the SECURITYGUARD file attribute. A guardfile also can be used to protect itself or other guardfiles.

A guardfile contains a list of users and programs that are authorized to access the guarded object along with each user's or program's corresponding access rights (e.g., read-only, write-only, read-write, execute, none). Execute access is allowed only on codefiles. Users are identified by USERCODEs or ACCESSCODEs while programs are known by file name and family name. A guardfile can specify both individual and group access rights based on USERCODEs, ACCESSCODEs and program names. A guardfile can also specify that user access is allowed only when running specific programs, and programs can only be allowed access when running under specific USERCODEs. The default access can be changed from none to read, write, read-write, or execute for users not specified in the guardfile.

A guardfile is created by the file owner using the GUARDFILE utility program. A user runs this utility through MARC, CANDE or WFL. Access rights are specified by the user in an input file which is processed by the utility to produce the guardfile. The contents of the guardfile can be displayed by MARC, CANDE or WFL.

The GUARDFILE utility creates the guardfile but does not attach it to the file being guarded. A separate action of attaching the guardfile is required to update the SECURITYGUARD attribute in the file header of the file being protected. The SECURITYGUARD attribute can be updated by using the WFL or CANDE SECURITY commands for disk files, the WFL VOLUME CHANGE command for tape volume families, and the Data and Structured Definition Language (DASDL) for the DMSII database. A second method of attaching a guardfile involves specifying the SECURITYGUARD file attribute in the file declaration of a program or in an executable program statement. A guardfile can be attached to one or more files by the owner or other users as long as the user has the proper title of the guardfile. Attaching a guardfile belonging to another user does not require access to the guardfile.

Attempts to open a guarded or controlled file cause MCP/AS to examine the access rules in the guardfile before granting or denying access. If the guardfile does not exist the file is protected as private. Otherwise the guardfile is searched sequentially for the first access right encountered that matches the USERCODE, ACCESSCODE, or program, or a combination of USERCODE, ACCESSCODE and program name. If no match occurs, the default access right is used. For example, USERCODE Fred is running program PAYROLL. If the guardfile contains read access for Fred then write access for the PAYROLL program, the access granted will be read. However, if the order in the guardfile is reversed then write access is granted. If neither Fred nor PAYROLL is identified in the guardfile and the default value is none, then no access will be allowed.

Tape Handling Mechanism

Unlike disk files which are secured on an individual basis, information placed on tape is secured at a tape volume family level. A tape volume is a single physical reel of tape. A tape volume family is a single tape volume or a set of tape volumes that have the same owner, volume name and file attributes. The volume directory database contains security information about each tape volume family and it is stored on disk. Information contained in the volume directory includes the volume name,

serial number, owner and security attributes. The tape volume serial number is a unique number used to index and locate this information in the volume directory.

Tape volumes are categorized as either complying or non-complying. Complying tape volumes have their serial numbers and security attributes entered into the volume directory and the tape label attributes match those recorded in the volume directory entry for the tape volume's serial number. The only exception to this rule is when the MATCHONLYSERIALNO attribute is set to true. In that case only the serial number of the tape volume must match the entry in the volume directory. The SAG recommends that the MATCHONLYSERIALNO attribute be set to false to ensure that security checks are made.

A tape volume is non-complying if the label attributes volume name, creation date, and creation site do not match the volume directory entry, if it is unlabeled, or if the serial number does not appear in the volume directory. Non-complying tape volumes can only be assigned by an operator using the ODT OU (Output Unit) or IL (Ignore Label) commands. The SAG instructs the operator to use discretion when issuing these commands.

The SECURITYTYPE, SECURITYUSE and SECURITYGUARD file attributes implemented for disk files are also used for tape volume families. Additional security attributes are FAMILYOWNER and PERMANENTLYOWNED. These security attributes are stored in the volume directory. The FAMILYOWNER attribute determines ownership of the tape volume family by USERCODE. Tape volume families may be unowned, temporarily owned or permanently owned. Unowned tape volume families are scratch tape volume families that are not permanently owned but may later become temporarily owned.

The PERMANENTLYOWNED attribute is a boolean value that indicates if the tape volume family has a permanent USERCODE assigned to it. When the PERMANENTLYOWNED attribute value is true, only files with a USERCODE matching the FAMILYOWNER attribute can be written to the tape volume family. Permanent ownership means that even if a tape volume family's contents are scratched, the user retains ownership and control of that tape volume family. Permanent ownership is established through the WFL VOLUME ADD statement. Only a security administrator, a privileged user or an operator may change permanent ownership of a tape volume family by deleting then re-adding the tape volume family in the volume directory.

If the PERMANENTLYOWNED attribute is false, tape volume family ownership defaults to the USERCODE of the first file written to the tape volume family. This tape volume family is known to be

temporarily owned. Files created by or associated with a different USERCODE cannot be created on a tape volume family once ownership has been established. However, if the family owner designates the tape volume family as public read-write, ownership will default to the USERCODE of the next user who creates a new first file on that tape volume family. Temporary ownership is also forfeited when the tape volume family is purged.

Tape volume family security attributes can be assigned in one of three ways. The WFL VOLUME ADD command is used to assign the security attributes when the tape volume family is initially defined to the system. The use of this command is limited to the Security Administrator, privileged users or operators. The WFL VOLUME CHANGE command can be used by a user to change only the SECURITYUSE, SECURITYTYPE and SECURITYGUARD attributes of the tape volume families the user owns. The last way of defining security attributes is by the tape volume family inheriting the security attributes of the first file created on a non-MATCHONLYSERIALNUMBER tape volume family.

Once the volume directory has been established, the security administrator uses ODT and WFL commands to maintain it. The WFL VOLUME DELETE/DESTROY command deletes the tape volume entry from the volume directory. The ODT "PG" (Purge) command makes the information on the tape volume family unavailable and identifies the entry in the volume directory as scratched. The ODT "TV MT" command is used to display the contents of the volume directory by tape volume families.

Each request to access a tape volume family forces the MCP/AS to compare the volume directory entry with the tape volume label. If MATCHONLYSERIALNO is true, only the serial number is compared. Otherwise the serial number, volume name, owner and creation site/date are used. If the directory entry and tape volume label do not match, an error message is displayed. Once a match has been made, the security attributes are examined and handled in the same manner as for disk files.

When a codefile is stored on a tape volume family by a user it becomes a data file. This is done by allowing ordinary users to only read and write files with a filekind of data or program symbol to the tape volume family.

Library Maintenance

A library maintenance tape is a collection of disk file images that are written to a tape volume by the MCP/AS LIBRARY/MAINTENANCE program. A user desiring to copy disk files

to a tape volume must have read access to the disk files and can only create a library maintenance tape under the user's own USERCODE. The system ensures that an image of the directory of disk files precedes the disk file images. Each file retains its disk file header which contains the USERCODE and security file attributes when copied from disk to tape. This information is not active while the files are on tape but it is preserved. When the files are restored to disk by the system, the disk file security information becomes active again.

Library maintenance tapes have a LABELKIND of library tape that identifies the tape volume as a library maintenance tape. MCP/AS will not allow the tape volume to be accessed by anyone other than the library maintenance routine. Library maintenance tapes belong to the system and are accessed via the MCP/AS LIBRARY/MAINTENANCE program only.

Logging

A Series MCP/AS audits security-relevant events and places audit information in the SYSTEM/SUMLOG file. User-specific and system-wide logging specifications control the events that are recorded. The events include subject and object creation, introduction, and deletion. The detailed list of these events is provided on page 49, "Audit."

A Series auditing information is generated internally by MCP/AS, the MCSs and the support libraries. Entries generated internally call the MCP/AS procedure WRITELOG directly. WRITELOG timestamps log entries and sends them to the system SUMLOG. Entries generated outside MCP/AS call MCP LOGGER, an MCP/AS procedure in the System Support Libraries. Entries may also be generated using the intrinsic, MCSLOGGER. The callers of MCP_LOGGER can include MCSs, system support Libraries, system processes, or privileged programs.

The System Data Access Support Library (SDASUPPORT) is used in retrieving and processing collected audit information. The records thus extracted contain the date and time of the event, user name, event type, success or failure of the event, the origin of request, and the name of the object operated upon, if any.

Internally each log record is made up of two parts: fixed and variable. The fixed part contains the job number of the associated job, the task number of the task causing the log

entry, current date and time, and the log entry type. The variable part is an extension of the fixed part containing task and file names, and other varying information.

The physical records of a single log entry are never split across two different logs, even when opening of a new log file was requested. Of course, entries for a single job or task may occur in more than one SUMLOG file.

A Series ensures that proper creation and protection of the SYSTEM/SUMLOG file is in effect. If the log file becomes nearly full (approximately 85%), the system prompts the operator to supply the system with more space for the log file. If the operator fails to perform such an action, the system will reboot, thereby removing jobs and freeing some disk space. The system may repeat this action as often as necessary.

Hardware/Stack Architecture

All A Series systems provide a single hardware domain of execution. Although in many systems, isolation of the TCB from user processes is provided by running the TCB in a completely separate (and privileged) hardware protection state, this is not true of A Series. Instead, a combination of capability-like hardware mechanisms and TCB software (including the compilers) is used to provide the necessary isolation. Because programs compiled by unprivileged users have no direct access to the machine instruction set or to the hardware enforcement mechanisms, the A Series TCB is able to isolate itself and other user processes from any attempted security violations. These capability-like hardware mechanisms are the tag architecture, the base and limit of stack registers, and the display registers.

This section describes in some detail the A Series Advanced System Architecture. The registers and mechanisms that support process isolation and stack integrity are also described.

Tag Architecture

The A Series architecture is a tag architecture which strictly separates code, control structures and data. One word in this architecture contains 52 bits. Bits 0-47 contain information while bits 48-51 are a tag for each word. TCB code and data structures are protected by these tags. Even-tag words are considered data. Odd-tag words denote control structures and code. Four bits allow sixteen (16) possible tag values. The current A Series architecture uses only tag values of 0-7. Tag values of 8-15 are reserved for later levels of the architecture.

Even-tag words may be single precision operands (tag-0), double precision operands (tag-2), bit vectors (tag-4) or uninitialized data (tag-6). A tag-6 word is used as an initial value of a variable. Operators (i.e., machine instructions) that expect an operand or descriptor will generate a hardware interrupt when a tag-6 word is encountered, thereby preventing the improper use of uninitialized data. Even-tag words are computational arguments rather than reference arguments or hardware control structures. The hardware does not permit the execution of even-tag words. Normal store operations are capable of storing over an even-tag word in memory. The normal store operations are: STOD (store delete), STON (store non-delete), and STAD/STAN (store delete/non-delete via address couple parameter).

Program code words, program and data control words, and memory address reference words have odd tags. Odd-tag words denote: indirect reference word (IRW, tag-1), code segment descriptor (CSD, tag-3), program code word (tag-3), mark stack control word (MSCW, tag-3), return control word (RCW, tag-3), top of stack control word (TSCW, tag-3), data descriptor (DD, tag-5), actual segment descriptor (ASD, tag-5) or program control word (PCW, tag-7). Since a tag-3 word may represent multiple structures, the intended structure is identified by context. Various types of reference words and descriptors are distinguished by bits within the tag-1 and tag-5 words. Odd-tag words are protected in actual memory. Normal store operations (i.e., STOD, STON, STAD, STAN) cannot write to odd-tag words. In order to write to odd-tag words it is necessary to write over the tag and data fields of the word. This is done by using the overwrite operators OVRD (overwrite delete) and OVRN (overwrite non-delete).

The tag field of words allows the hardware to differentiate between code, control structures and data. The hardware only executes code that has been retrieved from program code words, which are tag-3 words. Since the architecture allows the hardware to differentiate between code and data, execution of data is prevented. Further, code is executable only if it is produced by a compiler (see page 20, "Compilers"). Compilers must be explicitly identified and granted compiler privilege. This allows the A Series to limit and control an ordinary user's ability to create executable code.

Stacks

A Series systems are stack-based machines. Processes are represented by a family of stacks (i.e., one or more stacks). These stacks are considered the instantaneous state of that process.

Stacks are created through one of three language constructs: RUN, CALL and PROCESS. The RUN construct creates another stack which is independent of the current stack. MCSs utilize this construct to create independent stacks for new user processes. The CALL construct creates a dependent stack (child) that executes synchronously with the creating stack (parent). The PROCESS construct creates an asynchronous dependent stack. A dependent stack remains in the system only as long as the stack that created it. Stacks created by CALL or PROCESS belong to the same family of stacks as the stack that created them.

Since many stacks are active on the system, stacks are identified by a stack number. A stack number is an offset into the ASD table (see page 14, "Memory Management"). A stack contains a set of local addresses that comprise the process' current addressing environments. A set of local addresses within a stack is called an activation record or stack frame. Activation records directly correspond to procedures and blocks declared within a program. Each activation record contains two system words at its base: a Mark Stack Control Word (MSCW) and a Return Control Word (RCW). Both of these words are tag-3. The remainder of the words in an activation record are parameters or variables local to the procedure or block represented by the activation record.

The Mark Stack Control Word (MSCW) is the base word in an activation record and contains the history link and environment link. A history link points to the predecessor MSCW within the stack. History links always point to MSCWs in the same stack, while the environment links may point to another stack. At the head of the history chain is the 'F' register. This register always points to the most recent MSCW in the stack.

The environment link points to the MSCW at the base of the immediately global activation record. Environment links consist of ASD number and offset. Since an environment link contains an ASD number, it may point into another stack. Environment links only point to stacks within a family of stacks (i.e., a process). The chain of activation records created by environmental links forms an addressing environment, also known as an environmental chain. This addressing environment is determined by the declaration of a procedure or block within a program, not by a procedure's calling sequence. The lexicographic level of a procedure or block characterizes the block's nesting depth. The lexicographic level also identifies a procedure's or block's position within the environmental chain. A procedure would have a lexicographic level (i.e., nesting depth) one greater than the lexicographic level of the block that declared the procedure. The lexicographic level of the currently running procedure (the topmost activation record) is held in a register named 'LL'.

Along with the 'F' and 'LL' registers the A Series has a set of display registers (D[0] to D[15]) that point to the MSCWs in the process' environmental chain. A Series systems impose a limit of fifteen (15) lexicographic levels (i.e., the environmental chain may only contain fifteen activation records). This is a hardware limitation forced by the number of display registers available. This limitation does not affect recursive programs, since lexicographic levels are not affected by the sequence of procedure calls that lead to the invocation of a procedure.

Lexicographic level 0 is assigned to the global addressing environment of MCP/AS. The display register D[0] is the same for every process within the system.

Lexicographic level 1 contains the user's code segment dictionary which is maintained by the operating system. A code segment dictionary defines the code of a program. A dictionary is composed of code segment descriptors (CSD). A CSD identifies the number of code words in a segment and refers to an Actual Segment Descriptor (ASD). The reference to an ASD is an offset into the ASD table (see page 14, "Memory Management"). Programs can make only read and execute references to code and data defined in the code segment dictionary. Therefore many processes can share the same dictionary, read-only data, and code.

Lexicographic levels 2 through 15 occur in process stacks, which are the users' stack space. The base of each of these stacks is defined by the Base of Stack Register (BOSR); its top is bounded by another register, the Limit of Stack Register (LOSR). The item at the top of a process stack is pointed to by the 'S' register. In order to improve performance some A Series machines use two registers, 'A' and 'B', as the two logical, top-of-stack items. The absolute address of the base of every user stack is pointed to by an ASD.

Address References

Addresses within a stack always refer to words within virtual segments. An address may refer to a word in an activation record (in the same or another stack) or in another segment. References into another segment are pointers to an ASD and an offset into the segment defined by the ASD. References to words within a stack take the form of an address couple.

An address couple comprises a lexicographic level and an offset into the activation record (LL, Offset). Address couples always refer to an item in an activation record, and are the most common method available to refer to an address. They are the only form of address that occurs in program code files.

Process Isolation

Process switching, stack separation and stack growth are three functions relevant to process isolation in A Series. A discussion of these follows.

Process Switching

A "move to stack" (MVST) instruction is used when the system switches current executing processes. This instruction causes a "top of stack control word" (TSCW) to be written at the base of the currently active stack. The stack to be activated is identified and processor state is restored. Finally, the display registers are updated to the new addressing environment and execution continues.

The TSCW written at the base of the active stack specifies the height of the stack and supplies a link to the start of the historical chain. Processor state is restored by reading the TSCW and ASD of the inactive stack and setting the BOSR, LOSR, S, F, LL and D registers. The addressing environment is updated by traversing the environmental chain and setting the display registers. At this point the new stack is active and the original active stack is inactive.

Stack Separation

A Series treats stacks like any other segment. A stack is pointed to by an entry in the ASD table. Therefore, MCP/AS memory management is responsible for maintaining the separation of stacks, just as it is responsible for maintaining the separation of data segments or code segments (see page 14, "Memory Management").

As described earlier (see page 36, "Stacks") a stack may contain a pointer to an activation record in another stack. These pointers (i.e., environment links) are restricted to only refer to stacks within the same family of stacks. This restriction is enforced by the language definitions and compilers available to users.

Stack Growth

A stack can only increase in size by having data pushed on it, either as part of procedure entry or dynamically during procedure execution. The code generated for procedure entry fills all cells in the activation record with appropriate empty values (i.e., numeric zeroes, faulted descriptors) so that when the procedure begins execution, all local variables have defined values, and those values have no relation to the previous contents of the stack locations.

At some point a stack may attempt to grow beyond its assigned limit. A user stack is allowed some head room which is used to process a stack overflow. The head room is used to avoid running MCP/AS cleanup procedures in memory that the stack does not own. The following paragraphs discuss the possible actions that may occur when a stack overflows.

If MCP/AS is running on the stack and the stack has full head room (i.e., 287 words), the stack is enlarged by 256 words. This is accomplished by moving the limit of stack register (LOSR) into the head room of the stack. MCP/AS is then allowed to continue running. If a second stack overflow occurs a fatal system memory dump (i.e., a crash dump) is performed.

If the stack is running in control state (i.e., maskable external interrupts disabled) and has full head room, the LOSR is moved only 32 words into the head room of the stack. A clean point is identified in the stack and the stack is allowed to continue running. If the stack overflows before the clean point is reached, the process is terminated.

If the stack is running in non-control state and has full head room, the stack is enlarged by 256 words. The stack is not allowed to continue running. An MCP/AS entrypoint STACKSTRETCHER is called to locate a segment larger than the current segment in use by the stack. The stack is allowed to continue normal execution after STACKSTRETCHER completes.

If the head room of a stack is less than 255 words and a stack overflow interrupt occurs, the process is terminated. If the head room is less than 31 words then a fatal system memory dump occurs.

EVALUATION AS A C2 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 by both, and shall provide controls to limit propagation of access rights. 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

A Series provides discretionary access control between system subjects (processes running on behalf of users) and objects (i.e., disk files, tape volume families, printer files, card-reader files, communication files, DMSII databases). For a complete discussion of subjects and objects see page 6, "Subjects" and page 8, "Objects."

Access to objects can be either public, private, guarded or controlled. Public access allows any user to access the object. Private access allows only the object's owner (i.e., creator) to access the object. When created, objects are protected by default (i.e., private).

The owner of an object can selectively assign access rights to that object by creating a guardfile. A guardfile may contain access information that specifies the access rights of users, programs and ACCESSCODEs. Users, programs and ACCESSCODEs can be granted and denied access on an individual or group basis. The

access right of a user, program or ACCESSCODE that is not explicitly specified in the guardfile, is determined by the default value contained in the guardfile.

When using a guardfile to provide discretionary access control, the object being protected must be either guarded or controlled and the guardfile must be attached. For a guarded object, the owner is permitted access regardless of the guardfile's access list. When an object is controlled, even the owner's access rights are determined from the guardfile. An owner of an object creates a guardfile by using the SYSTEM/GUARDFILE utility program. Only the owner of an object can attach a guardfile to the object but the guardfile being attached need not be owned by the user. For a complete description of guardfiles see page 29, "Guardfile Mechanism."

Only the owner of an object can grant or deny access to that object. Once a user has access to an object that the user does not own, access to the object can not be given to any other user. The only exception occurs when write access to the guardfile is given. In that case, the owner must specifically give write access to the guardfile protecting the object. Since the owner controls who may access an object, and only the owner or a privileged user can pass this control to other users, propagation of access rights is limited.

Discretionary access control is provided for disk files by using a combination of security file attributes and guardfiles. (For an explanation of security file attributes see page 11, "Disk Management.") The owner of a disk file can deny all unprivileged users access to his file by declaring the SECURITYTYPE attribute as private. On the other hand, the file owner can allow other users read-only, write-only, read-write, or execute (for codefiles only) access by assigning the SECURITYTYPE attribute as public and appropriately setting the SECURITYUSE attribute. Guardfiles can be attached to disk files to allow a combination of access rights for different users, ACCESSCODEs and programs. As mentioned earlier, disk files will default to private if the SECURITYTYPE attribute is not specified.

Tapes are secured as a tape volume family in which all files contained within a tape volume family have the same owner. Tape volume family attributes are used in conjunction with guardfiles to provide discretionary access to a tape volume family. If a tape volume family is PERMANENTLYOWNED, another unprivileged user cannot create a new first file on the tape volume family even if the tape volume family is public read-write or the guardfile grants the user write access. A user can attach a guardfile to a

tape volume family that is temporarily or permanently owned by that user. For more information on tape volume families see page 30, "Tape Handling Mechanism."

The mechanisms for handling printer and punch-card files are identical; therefore, references made in this paragraph refer to both types of files. Direct access to printer FILEs (FILEs are the logical objects accessed by programs) cannot be accomplished by unprivileged users without operator intervention. Instead, printer files (files are a collection of data) are spooled to disk or tape by default. In the evaluated configuration, spooled files inherit the USERCODE of the user or task under which they are created. At the same time, the file is protected as private. Printer files could instead have guardfiles attached to them for additional flexibility.

In the evaluated configuration, card-reader FILEs are attached by default to the WFL compiler. Like printer FILEs, an unprivileged user cannot directly access a card-reader FILE. Instead, a card deck is spooled into a user's disk file by the WFL compiler where it is protected as a private file.

A DMSII database consists of a control file, a number of data files and an executable codefile called access control routines (ACR). The mechanism for providing disk file discretionary access controls is used to provide access control to the data files, control files, and ACR routines of a DMSII database. Separate guardfiles are attached to each of the files associated with the database. Read permission to the control file and execute permission to the access control routines are necessary for users desiring to access the database through the access control routines. To ensure that users only access the data files using the DMSII database, a guardfile must be attached to the data files to deny direct read or write access. The discretionary protection provided on the three database files is capable of requiring users to access database information through the access control routines. The evaluation documented by this report made no attempt to examine access control provided by the access control routines.

A Series implements three different types of communication FILEs: ODT FILEs provide communication with Operator Display Terminals; Remote FILEs provide communication with user stations over datacomm lines; port FILEs provide communication between tasks.

To provide communication between an ODT and a program, ODT FILEs (also called SPO FILEs) are implemented. ODT FILEs can be opened for input, output, or input and output. A request made from a program initiated at an ODT to open an ODT FILE for output causes

the FILE to be attached to that ODT only. In the evaluated configuration, ODT FILEs opened in any other manner require operator intervention.

Attaching a FILE to a remote station is moderated by the controlling MCS. In the evaluated configuration, the CANDE option LAISSEZFILE is set to zero. To attach a remote ODT FILE, a user must be logged onto a station. The user is notified of a foreign FILE attempting to communicate with that station. The user has the option to deny or allow the connection. Each station is limited to one remote input FILE at a time, although it may have multiple output files.

Port FILEs provide inter-program communication between independent tasks. Port-subfiles (also called subports) allow a task to concurrently communicate with two or more other tasks through a single port file. The principal attributes used to match port FILEs are FILENAME, MYNAME, YOURNAME, SECURITYTYPE and YOURUSERCODE. All subports must have the same SECURITYTYPE but the remaining attributes may be set on an individual subport basis.

In order to establish a port FILE, tasks must agree upon a FILENAME. Once this match has occurred, the MYNAME and YOURNAME attributes are used together to provide unique identification at the subport level. Each task requesting the port FILE must identify the value of MYNAME so that the other task requesting the port FILE will use that value for the YOURNAME attribute. For example, if task A wishes to establish a port FILE with task B, MYNAME and YOURNAME attributes for task A might be Reno and Tahoe, while the values for task B's attributes would be Tahoe and Reno, respectively.

The SECURITYTYPE attribute for each subport may be set to either private (default) or public. When the SECURITYTYPE attribute offered by both tasks is private, the matching tasks must be running under the USERCODE specified by the YOURUSERCODE attributes. If the SECURITYTYPE value is public, no USERCODE constraint is imposed. If only one of the subports is private, then security checking is done in one direction only.

The guardfile mechanism implemented in A Series provides the functionality of an access control list that is capable of specifying both a list of users and their access rights, and groups of users and their access rights. Access rights are specified as read, write, read-write, execute and no access.

Conclusion

A Series MCP/AS Release 3.7 satisfies the C2 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

A Series MCP/AS Release 3.7 satisfies(1) the B3 Discretionary Access Control requirement. A Series guardfiles can include an arbitrary number of USERCODEs, ACCESSCODEs and program names, each of which can be either granted or denied access.

Object Reuse

Requirement

> All authorizations to the information contained within a storage object shall be revoked prior to initial assignment, allocation, or reallocation to a subject from the TCB's pool of unused storage objects. No information, including encrypted representations of

---

(1) Although A Series MCP/AS Release 3.7 satisfies this requirement at the B3 level, it does not satisfy the assurance requirements above its rated level.

> information, produced by a prior subject's actions is to be available to any subject that obtains access to an object that has been released back to the system.

Applicable Features

In A Series, MCP/AS performs several functions which address the object reuse requirement. MCP/AS writes zeros into all main memory locations requested by a task. Upon demand, I/O buffers are allocated by MCP/AS in main memory. They, too, are cleared upon allocation.

All of main memory is overwritten with zeros during system initialization. Stack frames, which are implemented in main memory, are handled this way during system initialization preventing data reuse at startup.

Local addressing spaces for procedures are activation records (see page 36, "Stacks") that are initialized upon their creation (procedure entry). Upon procedure entry, the processor registers that determine the procedure's addressing environment are set. During process switching, all processor registers are either filled as described on (see page 39, "Process Switching") or recalculated.

A stack can only increase in size by having data pushed on it, either as part of procedure entry or dynamically during procedure execution. The code generated for procedure entry fills all cells in the activation record with appropriate empty values (i.e., numeric zeroes, faulted descriptors) so that when the procedure begins execution, all local variables have defined values, and those values have no relation to the previous contents of the stack locations.

The DISKSCRUB option prevents the new area of a disk file from being read until that area has been overwritten by the user or MCP/AS. Setting the DISKSCRUB option forces MCP/AS to perform an overwrite of the disk space during reallocation. The DISKSCRUB option must be set at system initialization in the evaluated configuration. Additionally, if the SENSITIVEDATA file attribute is set the disk space of a file is overwritten upon deallocation. Therefore, if both features are invoked, two discrete overwrites are done.

Data on labeled tape volume families can only be read between the logical beginning and end of tape and only after the discretionary access control checks are made. When a tape file is created, a double tape mark is used to indicate the end of

accessible data. In this way, the system prevents a user from reading beyond the end of accessible data. A user may only perform logical reads and writes to tape volume families. This prevents users from reading tape areas before they have been written. When a tape volume family is labeled as scratched, the system will not permit it to be read.

Within A Series, communication FILEs are treated uniformly as I/O buffers. Communication FILEs are zero-filled when allocated.

Conclusion

A Series MCP/AS Release 3.7 satisfies the C2 Object Reuse 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 use a protected mechanism (e.g., passwords) to authenticate the user's identity. 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

In A Series, a station is logically connected either to COMS or CANDE. If the station is connected through COMS, the SECADMIN must designate MARC as the default window, which acts as the operator/user interface. A user must login with a valid USERCODE and password before any other action is permissible.

Terminals connected to CANDE are also required to successfully login with a valid USERCODE and password. CANDE has LOGIN station options, ALLLOGIN and DIALLOGIN, which require users to be authenticated before any remote file can be opened at that

station. For both CANDE and COMS, passwords are entered in a video blanked (echo-suppressed) field to prevent their disclosure.

The HELLO command can be used to end the current session, without disconnecting dial-up lines. When HELLO is entered, the current session ends, and CANDE displays resource usage statistics for the previous user, and a welcoming message for the next user. The station becomes ready for the next user to enter a valid USERCODE and password. In the evaluated configuration, all USERCODEs have a password associated with them.

User authentication data is saved in the USERDATAFILE, which is a protected file. The USERDATAFILE in use by the system is owned by the SECADMIN as a private, system file. Access to this file is limited to the SECADMIN. The USERDATAFILE may only be accessed through the use of DCALGOL functions. The ACCESSRESTRICTION bit in the USERDATAFILE's header is checked during FIBOPEN to prevent direct access to the file by an unauthorized user.

In A Series, it is possible to uniquely associate a user with either one USERCODE or one ACCESSCODE. One possible configuration would uniquely associate individual users with one ACCESSCODE, and many users would share a USERCODE. This configuration would provide grouping by USERCODEs and individual accountability by ACCESSCODEs. This is not acceptable in the evaluated configuration since the evaluated mechanisms of A Series provide individual accountability through the use of USERCODEs. In the evaluated configuration each individual user must be associated with one and only one USERCODE, preserving individual accountability. The association of an individual user with one USERCODE allows that USERCODE to uniquely identify the user. Many users may share the same ACCESSCODE, thereby forming a group. This association of one user to one USERCODE is necessary in the evaluated configuration. This distinction is clearly explained in the Security Administration Guide.

As each task is being initialized, it is assigned a mix number. A mix number is a four digit number that is unique among active tasks. Mix numbers may be reused, but MCP/AS ensures that no two tasks that are currently active will be assigned the same mix number. Once a mix number has been assigned to a new task, an audit record is generated that associates the mix number with the USERCODE. Every audit record contains the mix number of the task responsible for the action. This enables the TCB to uniquely identify the user that is responsible for the action.

Conclusion

A Series MCP/AS Release 3.7 satisfies the C2 Identification and Authentication 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 (e.g., file open, program initiation), deletion of objects, actions taken by computer operators and system administrators and/or system security officers, and other security relevant events. 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. The ADP system administrator shall be able to selectively audit the actions of any one or more users based on individual identity.

Applicable Features

Audit Event Specification

A Series MCP/AS has the capability to audit all occurrences of security-relevant events by placing audit information in the SYSTEM/SUMLOG file. User-specific and system-wide logging specifications control the events that are recorded in

SYSTEM/SUMLOG. User-specific events are attributable to individual users and include successful actions, failed actions, security-relevant actions, and security violations.

System-wide logging specifications are very extensive. They are categorized, by major event types, into several types of records including:

- identification,
- job,
- error and status messages,
- MCS,
- miscellaneous,
- MultiLingual System (MLS) message,
- file status,
- Datacomm configuration,
- COMS configuration.

Each major event type may have several minor event types further specifying the type of event logged.

Auditable security-relevant events caused by all types of users include:

- start and end of job and task,
- file and database open and close,
- library link and delink,
- system response to selected commands,
- MCS login, logout, message record, security violation, station application, and window open and close,
- halt/load of the system,
- log file release,
- SETSTATUS invocation,
- general security violation,
- controller commands,
- print subsystem commands,
- installation of and changes to USERDATA,
- primitive commands,
- ODT commands,
- MLS RSVP, INFO, unit RSVP, and special RSVP messages,
- file creation, removal, rename, and security attribute change,
- installation of a new datacomm configuration file and datacomm IDC changes,
- changes to COMS configuration file.

Any of the above events can be used as a criterion for selective logging.

Audit Log Analysis

A Series provides three utilities for examining SYSTEM/SUMLOG: the LOGGER, the System Management Facility II (SMFII) and LOGANALYZER. The LOGGER and the SMFII are primarily used in gathering system usage statistics. The LOGANALYZER is the utility providing the most detailed audit log analysis.

The LOGANALYZER is capable of extracting records from either the current or previous logs which were generated during a specified time interval. The extracted records can additionally be grouped according to specified subjects. The LOGANALYZER can be directed to extract records containing information pertaining to USERCODE, ACCESSCODE, COMS, DMSII operation, file operations, datacomm configuration operations, Library linkages, operator functions, operation result, and security-related operations.

The resulting records contain all the required information: date and time of the event, user identification, event type, success or failure of the event, the origin of request, and the name of of the object operated upon, if any (see page 33, "Logging").

Audit Log Protection

A Series ensures proper creation and maintenance of the SYSTEM/SUMLOG file. SYSTEM/SUMLOG is created as a private file owned by the system and all access to it is regulated by the InfoGuard System Data Access (IGSDA) support Library. IGSDA is a privileged library that filters user access to SUMLOG files and is required in the evaluated configuration. IGSDA allows ordinary users to view log records produced by the actions the user took on the system, except for security violation records. An ordinary user can also view public records, which include maintenance and halt/load records.

When the current SYSTEM/SUMLOG file becomes nearly full and LOGHANDLER is unable to allocate any additional log storage, a warning is posted on the ODT advising the operator of the situation and requesting additional space. While LOGHANDLER is waiting for space, no new tasks are created. This is done to reduce the amount of subsequent loggable events.

If the system runs out of its log space before the operator has a chance to free some disk space, the system will reboot (halt load). After the reboot, LOGHANDLER will either be able to find the additional space (perhaps released when temporary files are

deleted) or task selection will be stopped, again allowing the operator to respond. Eventually the system may perform no useful work but the integrity of its audit log remains uncompromised. As new log files are opened, all active log information is retained, thereby providing a continuity of logging.

Conclusion

A Series MCP/AS Release 3.7 satisfies the C2 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). Resources controlled by the TCB may be a defined subset of the subjects and objects in the ADP system. The TCB shall isolate the resources to be protected so that they are subject to the access control and auditing requirements.

Applicable Features

A Series TCB is the MCP/AS, system libraries, compilers, message control system and privileged programs. The subjects in A Series are jobs and tasks (i.e., stacks executing on behalf of a user). The objects in A Series are disk files, tape volume families, printer files, card-reader files, communication files and databases.

All processors in the A Series product line provide a single state for execution. In order to provide a self-protecting domain of execution for the TCB and a user domain of execution, A Series system software must strictly control the creation of executable code (see page 20, "Compilers"). For this reason, only UNISYS-supplied compilers can be loaded in the evaluated system.

The A Series TCB (specifically MCP/AS) uses the file attribute filekind to ensure that all executable code has been created by installed compilers (see page 11, "Disk Management"). The hardware enforced tag bits and the filekind attribute together prevent the system from executing data. The filekind attribute

and FIBOPEN mechanism prevent users from writing into codefiles, while the user non-addressable tag bits provide stack integrity checks. The tag bits, filekind attribute and FIBOPEN mechanism together prevent users from writing their own compilers. The responsibility for codefile integrity rests upon these three mechanisms, as well as on the conformance of a compiler with its language specification, and the process isolation provided by MCP/AS.

While compilers are not responsible for making access checks on files, compilers are responsible for ensuring that all accesses to a file are through valid descriptors. A valid descriptor is created by the MCP/AS procedure FIBOPEN. Compilers are responsible for causing a file to be opened by MCP/AS before it may be accessed. Therefore, MCP/AS makes the access checks at run-time as a file is being opened.

Files in the A Series are located through the use of the FLAT, the FAST and the PAST (see page 11, "Disk Management"). A user's access to a file is computed from information located in the FLAT entry of a file when the file is initially opened by a task (i.e., the first reference to a descriptor for the file). When a file is opened, an audit record is generated that identifies the task attempting to open the file.

As stated on page 16, "Process Control", each task, as it is initialized, is assigned a mix number. Since a job in execution is a task, jobs are also assigned mix numbers when created. A mix number is a four digit number that is unique among active tasks. Mix numbers may be reused but MCP/AS ensures that no two currently active tasks have the same mix number. Once a mix number has been assigned to a new task, an audit record is generated that associates the mix number with a USERCODE. Every audit record contains the mix number of the task responsible for the action. Therefore, the mix number can be used to uniquely identify the user responsible for the action.

Conclusion

A Series MCP/AS Release 3.7 satisfies the C2 System Architecture requirement.

Additional Requirement (B1)

The following changes are made in this requirement at the B1 level:

> ADD: The TCB shall maintain process isolation through the provision of distinct address spaces under its control.

Conclusion

A Series MCP/AS Release 3.7 satisfies(1) the B1 System Architecture requirement. The A Series maintains process isolation by providing users with distinct address space in the form of stacks. Further these stacks are bounded by base of stack and limit of stack registers. Stacks and processes are described in more detail on page 36, "Stacks" and page 16, "Process Control."

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

UNISYS provides a set of maintenance diagnostics that can be used to verify the correct operation of all system hardware and firmware. The use of these diagnostic tests and tools assures the Administrator that the hardware is working properly and meets the A Series specifications. System diagnostics allow all internal paths and registers to be exercised and tested in a predictable manner.

---

(1) Although A Series MCP/AS Release 3.7 satisfies this requirement at the B1 level, it does not satisfy the assurance requirements above its rated level.

Verification of the compilers on A Series is performed by the use of SYSTEST/LANG, which is a set of diagnostics and validation routines. These routines test the compilers and ensure the correct operation of the compilers. The use of SYSTEST/LANG assures the Administrators that installed compilers work as specified.

Conclusion

A Series MCP/AS Release 3.7 satisfies the C2 System Integrity requirement.

Security Testing

Requirement

> The security mechanisms of the ADP system shall be tested and found to work as claimed in the system documentation. Testing shall be done to assure that there are no obvious ways for an unauthorized user to bypass or otherwise defeat the security protection mechanisms of the TCB. Testing shall also include a search for obvious flaws that would allow violation of resource isolation, or that would permit unauthorized access to the audit or authentication data.

Applicable Features

The NCSC evaluation team performed testing of A Series in July 1987. The system tested included an A3 computer (with two disk drives, two tape drives, a printer and five terminals) running A Series MCP/AS Release 3.7.

The evaluation team began testing by first loading and generating the system from the distribution tapes. Following instructions in the Security Administration Guide, the team configured the system for the C2 mode of operation. The team then executed the entire set of vendor tests. There were approximately 1,900 tests and a vast majority of these were fully automated. A Series passed all the tests.

The vendor tests were further supplemented by several tests developed and executed by the team. They included: checking for the correct access to newly-created system and system files, placing conflicting access information in guardfiles, forcing the

audit log to exhaust its allotted disk space, attempting to read data from newly-created objects, attempting to modify file types, and attempting to modify various control blocks using a system debugging tool. All of the team tests were correctly handled by the system.

It should be noted that the development of the vendor's tests located six (6) implementation flaws in A Series. All these flaws were corrected prior to the team testing.

Conclusion

A Series MCP/AS Release 3.7 satisfies the C2 Security Testing 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

The A Series Security Features Operations and Programming Guide, dated July 1987 (document # 1195203), is directed at ordinary users and it provides detailed information about the available A Series security features.

The Security Features Operations and Programming Guide (SFOPG) includes descriptions of A Series security features followed by instructions on their use. The SFOPG covers the pertinent areas of user login, password manipulation, user classes and attributes, disk file and tape volume family access and protection, program and library access and protection, and database access and protection.

In addition, the SFOPG contains a set of examples corresponding to most features previously described. As needed, references to other system manuals are provided.

Conclusion

A Series MCP/AS Release 3.7 satisfies the C2 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.

Applicable Features

The A Series Security Administration Guide, dated July 1987 (document # 1195195), makes suggestions to the security administrator on how to operate the system in a low, medium, or high security configuration. A high security configuration refers to the InfoGuard security enhancements with the CLASS option set to S2.

The Security Administration Guide (SAG) consists of two main parts. The first part provides an overview of the security mechanisms and the protection philosophy. Log-on policies, ACCESSCODE privileges and password protection mechanisms are described. Also included are warnings and recommendations for controlling system privileges. The chapter on auditing describes the auditing tools, events that are audited and the functions available to read the audit logs. This includes both recommendations and examples of the most useful items to audit. This document contains a discussion on how to provide grouping of users and recommends an initial configuration.

The second part of the SAG contains detailed instructions for USERCODE management by describing the MAKEUSER utility. Topics that are discussed include defining USERCODEs, associating privileges with those USERCODEs, and protecting the USERDATAFILE.

Specific guidance is provided in an Appendix to assist the security administrator in establishing and running A Series in a C2 environment using the InfoGuard security enhancements.

Conclusion

A Series MCP/AS Release 3.7 satisfies the C2 Trusted Facility Manual requirement.

Test Documentation

Requirement

> The system developer shall provide to the evaluators a document that describes the test plan, test procedures that show how the security mechanisms were tested, and results of the security mechanisms' functional testing.

Applicable Features

UNISYS Corporation provided a comprehensive list of security mechanisms. This list was identified and verified for completeness through an extensive and iterative review of all the system functions. Subsequently, A Series security mechanisms tests were developed and they are documented in the Burroughs A Series Security Evaluation: Security Mechanisms manual. This manual contains a section, Test Requirements, defining the test mechanisms and the approach to testing. In addition, there are twenty six (26) sections, one for each security mechanism, providing a functional description of each mechanism, a functional test overview, functional test scenarios, and expected test results.

Each of the functional test scenarios has a corresponding document containing the actual test description and test code to be run for that scenario. The code contains a scenario overview and well-commented test cases. A majority of the tests are automated. Each includes the necessary setup of its environment, a detailed description of what is being tested, and a test result file.

UNISYS Corporation has included these tests in the overall system testing package fully intending to use them and to maintain them in their system development efforts.

Conclusion

A Series MCP/AS Release 3.7 satisfies the C2 Test Documentation requirement.

Design Documentation

Requirement

> Documentation 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. If the TCB is composed of distinct modules, the interfaces between these modules shall be described.

Applicable Features

The A Series software domain philosophy is described in the E-Mode Usage, A Series Security Architecture, and E-Mode Operator Set documents listed below. These documents are targeted at developers of system software (e.g., compilers, MCP/AS, support libraries).

The A Series Security Architecture identifies the protected objects and briefly describes the protection mechanisms. The A Series Security Architecture, and design review documents, explain to a system developer the functionality of MCP/AS, the auditing mechanism, the identification and authentication mechanisms, and other system functions.

- E-Mode Usage, Darrell High, 15 July 1986.
- A Series Security Architecture, Darrell F. High, September 1986.
- Burroughs Large System MCP Manual Release 3.5, A Series, Burroughs Corporation, Document Number 48232.
- Functional Specification of the E-Mode Operator Set (Level Beta), The Burroughs Corporation SDS 2358-7157, Revision F2, March 10, 1986.
- The entire set of UNISYS Design Review Documents.

Conclusion

A Series MCP/AS Release 3.7 satisfies the C2 Design Documentation requirement.

EVALUATORS' COMMENTS

Object Reuse Flexibility

UNISYS in their A Series line of products has exceeded the Criteria requirement for object reuse. A Series provides two different mechanisms which perform object reuse. A system-wide mechanism clears objects as they are allocated, and a per disk file mechanism clears disk files upon deallocation. The two mechanisms, DISKSCRUB and SENSITIVEDATA, work independently yet complement each other. The two different mechanisms provide considerable flexibility. User with sensitive information have assurance that their information is cleared when their file is deleted.

A Series TCB Description

UNISYS performed a study of A Series system software. This examination determined the type of interface and nature of trust for software they provide. The nature of trust placed in a system software codefile, MCP/AS procedure, or system library entrypoint was identified as one of four general classes. These four general classes are protect integrity of memory objects, implement security policy correctly, use privilege judiciously (do not abuse), and differentiate data for independent users properly. This study is documented in an internal UNISYS document entitled A Series Trusted Computing Base (TCB) Description. This study is a valuable tool that will enhance the corporation's ability to maintain A Series securely.

Password Management

A Series can be configured such that the system generates random pronounceable passwords 7 to 11 characters long for all users. When A Series is configured in this way, users are not permitted to select their own passwords.

A Series can also be configured to expire a user's password. Password aging is enabled by administrators on a per-USERCODE basis. A distinct aging period must be specified for each USERCODE that has password aging enabled. As a result of

password aging, the system will force new password assignment when the old password has expired. This option applies only to USERCODE passwords; ACCESSCODE passwords are not affected.

A user's password is not abruptly expired by this mechanism. Instead passwords change from an active state to a warning state to an expired state. Users with passwords in the warning state receive a message whenever they login or initiate a task/job. This message indicates the number of days left before expiration. Users with expired passwords are forced to change their password before they are permitted to do any other action on the system. This password aging and generation mechanisms provide a flexibility that will enhance a security administrator's ability to operate a system securely.

Configuration Management

The UNISYS Corporation has a process to keep track of changes made to the software that is delivered for A Series systems. They refer to changes as patches. Whenever it is determined that a patch is needed to improve or correct the system software, it is recorded in a database management system called PATCHMANAGER. Patches, their testing and their documentation are the result of programmers responding to a need to change the software. One programmer develops, tests and documents the patch that solves the problem. Another programmer then audits the code, test method and documentation. Discrepancies are resolved by the two programmers. Lastly, the responsible manager reviews the audited code with respect to the original need. The manager is the one held accountable and responsible for the correctness and completeness of the patch and its installation in the released code.

Real-time Security Alarms

Any station connected to MCP/AS can be defined as a security station. As such, it can receive announcements of all MCP/AS and MCS security violations as they occur on the system. These announcements are displayed on the station regardless of other activities that may be taking place at the station. The messages include the time of the violation, a description of the violation, the subject, the object, and in some cases the station identifier.

This feature can allow a system administrator to become rapidly aware of any security violations. It allows the administrator to play an active role in thwarting attempts to circumvent system security.

Selective Auditing and Reporting

Rather than being limited to auditing all events, MCP/AS can be instructed to perform selective auditing. This type of auditing can be directed to all successful actions, all failed actions, and all security violations for particular USERCODEs. This approach allows for an efficient auditing of specified individuals. Such a feature is desirable because it both reduces the amount of logged information requiring processing and improves system performance, thereby encouraging administrators to use auditing tools.

In cases where large amounts of audit information need to be processed, LOGANALYZER and SMFII can be used to produce tailored reports containing detailed analyses of information extracted from the audit logs. These reports can span specific time periods and users. They can also consolidate audit records and present counts of violations.

Tape Volume Family Security

A Series provides an automated method of handling magnetic tapes instead of a more typical procedural method that requires operator intervention during tape assignment. Once a tape volume family has been mounted, MCP/AS checks the volume directory database to determine if a user has the proper access rights to the tape volume family. After security checks are made and the user has the proper access, the system automatically assigns the tape volume family to the user. Operator intervention is not required.

Closing Comments

We would like to express our satisfaction in working with UNISYS. Throughout the evaluation UNISYS provided the NCSC evaluation team with excellent support, including customized training specifically designed to address relevant topics, system access to allow additional hands-on experience, a convenient testing

schedule and system availability, and very timely responses to team questions. People at UNISYS - specifically those at Burroughs in Lake Forest, and those at SDC in Santa Monica and McLean - demonstrated an outstanding level of professionalism while being involved in this effort.

EVALUATED HARDWARE COMPONENTS

The hardware covered by this evaluation is the entire current A Series Advanced System product line. The primary requirement for hardware evaluation is that the hardware function properly. This is verified by the system integrity tests (see page 54, "System Integrity") and was not given a detailed re-evaluation by the team. The integrity assurances provided by the UNISYS-supplied diagnostic tests are satisfactory.

List of Evaluated Components

This appendix lists, by category, the UNISYS identification numbers for all hardware components that are covered by this evaluation. To operate in correspondence with the C2 rating, the hardware configuration of an installation must contain only components listed in this section.

Central Processing Units

The models noted in this attachment by 'X' are MCP/AS only systems and superseded the non-X models, but that is the primary distinguishing factor. All of the following models identified fully support MCP/AS.

A 1	
A 2	
A 3	models D, F, K
A 4	
A 5	model F
A 6	
A 9	models DX, FX, D, F
A 10	models DX, FX, HX, D, F, H
A 12	
A 15	models FX, HX, IX, JX, KX, LX, MX, NX, F, H, I, J, K, L, M, N

The CPU hardware identified here can be ordered with varying amounts of memory and different DLPs, depending on site equipment needs. The underlying hardware, while it has different components and differing microcode inside a particular sub-system

or module, presents a uniform and consistent interface to the system software across the models listed above. The following is hardware available from UNISYS at the time of this evaluation.

I/O Subsystem Modules

Style Number	Description
AX 341-90	Operator Display DLP-3
AX 110-90	Card Reader DLP-2
AX 393-90	NRZ Magnetic Tape DLP-3
AX 395-92	GCR/PE Magnetic Tape DLP-3
AX 395-91	PE Magnetic Tape DLP-2
AX 112-90	Card Punch DLP-2
AX 246-92	Line Printer DLP-2 2000 LPM
AX 247-94	Train Printer DLP-2AX Non-impact Printer DLP-3
AX 304-90	Disk Pack DLP-2 Interlace
AX 304-91	Disk Pack DLP-3 Sequential/Interlace
X 110-90	BCL Card Reader DLP
X 112-90	BCL Card Punch DLP
X 246-96	Printer/Tape DLPII
X 293-30	Non-impact Printer DLP
X 393-90	NRZ Magnetic Tape DLP-3
X 395-92	GCR/PE Magnetic Tape DLP-3
X 395-91	PE Magnetic Tape DLP-2
X 304-[illegible]	Host Transfer Sequential/Interlace DLP
X 304-95	SMD DLP II
X 304-97	XSMD DLP
X 378-10	Data Communication DLPII
X 304-90	Host Transfer Interlace DLP
X 246-92	Printer DLP (B 9246-21)
X 246-92	Printer DLP (B 9246-10/12)
X 901	Non-FCC DLP to FCC Upgrade Kit #1
X 902	Non-FCC DLP to FCC Upgrade Kit #2
X 903	Non-FCC DLP to FCC Upgrade Kit #3
X 904	Non-FCC DLP to FCC Upgrade Kit #4
A 369-10	RS232 Character/Bit-Oriented Interface
A 369-11	CCITT Character/Bit-Oriented Interface

A 369-12	TDI Character/Bit-Oriented Interface
A 369-41	Auto Call Unit II
X 394-93	FIPS HYPC DLP-2 For A3/A9/A10/A12
A 12-IIO	Independent I/O Cabinet with 2 Bases (1 DC, 1 MC, 1 BC Per Base)
A 12-IIO	Additional Independent I/O Cabinet With 2 Bases (1 DC, 1 MC, 1 BC Per Base)
X 320-2	ISC Host DLP/B974 (For A15IIC/II2)
A 378-1	Line Support Processor III (LSPIII)
A 378-3	Quad Character-Oriented Line Adapter 2
A 378-4	Quad Bit-Oriented Line Adapter 2

Disk Products

Style Number	Description
B 9484-5/51	(206) 2 Spindles 130.4MB
B 9484-12	(677) 1 Spindle 3 Phase AC Power 252MB Interlaced 242MB Sequential
B 9484-12G	(677) 1 Spindle Single Phase AC Power 252MB
B 9484-13	1 Spindle 3 Phase AC Power 252MB Interlaced 242MB Sequential
B 9484-13G	1 Spindle Single Phase AC Power 252MB
B 9494-4/-41	(207) 2 Spindles 402 MB
B 9494-10I	(659) 2 Spindles 1084 MB 600 KB/Second
B 9494-10S	(659) 2 Spindles 962 MB 1.2 MB/Second
B 9494-12	(3680) 1 Spindle 2 Actuators 868 MB/Spindle 434 MB/Actuator, 3 MB/Second Transfer Rate
MD4-2	2 Modules 245.6 MB
MD4-4	4 Modules 491.2 MB
MD8-2	2 Modules 500 MB
MD8-4	4 Modules 1000 MB

Disk Controllers

Style Number	Description	Disks Controlled
B 9387-41	Single data path controller up to 8 spindles (1x8)	B 9484-5/51 (206) B 9494-4/41 (207) Only
B 9387-42	Double data path controller up to 8 spindles (2x8)	B 9484-5/51 (206) B 9494-4/41 (207) Only
B 9387-51C	Basic 1 x 8 Controller	
B 9387-52C	Basic 2 X 8 Controller	
B 9387-99B	BLT Interface - for B 920/ B 930/ CP 9500 (B 9387-51C Only)	B 9484-5/51 (206) B 9494-4/41 (207) Only
B 9387-99D	HT DLP interface for A3, A9 Large Systems with B X304-90 DLPs	(B X304-90 DLP) B 9484-5/51, B 9494-4/41 and B 9484-12/13 (interlaced only) B 9494-51, B 9494-101
B 9387-99S	BLTA Sequential/Interlaced interface - A9, A15 with B X304-91 DLPs	B 9484-5/51, B 94949-4/41, B 9484-12/13, B 9494-51/5S, B 9494-101/10S
B 9387-24	Disk exchange 4 x 16 for use with two 2 x 8 controllers	Based on Controller
B 9387-26	Disk exchange 6 x 16 for use with three 2 x 8 controllers	Based on Controller
B 9387-28	Disk exchange 8 x 16 for use with four 2 x 8 controllers	Based on Controller

B 9387-34/36/38	Disk exchanges 4 x / 6 x / 8 x 16	Based on Controller
B 9389	Dual storage Controller for A9, A15 systems with B X304-91 DLPs - connects to B 9399 Dual String Controller	
B 9399	Dual String Controller - attached to B 9389 and B 9494-12 Disks - up to 8 - B 9494-12s	

Magnetic Tape Products

Style Number	Description
B 9498	PE Magnetic Tape Streamer
B 9495-2/82	75 IPS 120KB TR Rate MTU-PE (Also with NRZ option)
B 9495-3/83	125 IPS 200KB TR Rate MTU-PE (Also with NRZ option)
B 9495-8/88	50 IPS 80KB TR Rate MTU PE Only
B 9495-22	75 IPS GCR/PE MTU 470/120KB TR Rate
B 9495-23	125 IPS GCR/PE MTU 780/200KB TR Rate
B 9495-32	75 IPS GCR/PE MTU 470/120KB TR Rate
B 9495-33	125 IPS GCR/PE MTU 780/200KB TR Rate
B 9495-24	200 IPS GCR/PE MTU 1250/320KB TR Rate
B 9495-32M	75 IPS GCR/PE MTU with built-in 1x4 Formatter
B 9495-33M	125 IPS GCR/PE MTU with built-in 1x4 Formatter

Tape Controllers

Style Number	Description	Tape Units Controlled
B 9499-14S	One Channel, up to 4-50 IPS MTU (1x4) PE Master Electronic Controller (MEC)	B 9495-8, B 9495-88
B 9499-14M	One Channel, up to 4-75 IPS MTU (1x4) PE Master Electronic Controller (MEC)	B 9495-2, B 9495-82
B 9499-14H	One Channel, up to 4-125 IPS MTU (1x4) PE Master Electronic Controller (MEC)	B 9495-3, B 9495-83
B 9499-18S	One Channel, up to 8-50 IPS MTU (1x8) PE Master Electronic Controller (MEC)	B 9495-8, B 9495-88
B 9499-18M	One Channel, up to 8-75 IPS MTU (1x8) PE Master Electronic Controller (MEC)	B 9495-2, B 9495-82
B 9499-18H	One Channel, up to 8-125 IPS MTU (1x8) PE Master Electronic Controller (MEC)	B 9495-3, B 9495-83
B 9499-28S	Two Channels, up to 8-50 IPS MTU (2x8) PE Master Electronic Controller (MEC)	B 9495-8, B 9495-88
B 9499-28M	Two Channels, up to 8-75 IPS MTU (2x8) PE Master Electronic Controller (MEC) (Also with NRZ Control Modification for one channel)	B 9495-2, B 9495-82
B 9499-28H	Two Channels, up to 8-125 IPS MTU (2x8) PE Master Electronic Controller (MEC) (Also with NRZ Control Modification for one channel)	B 9495-3, B 9495-83

B 9499-2XM	Two Channels, up to 16-75 IPS MTU (2x16) PE Master Electronic Controller (MEC) (Also with NRZ Control Modification for one channel)	B 9495-2, B 9495-82
B 9499-2XH	Two Channels, up to 16-125 IPS MTU (2x16) PE Master Electronic Controller (MEC) (Also with NRZ Control Modification for one channel)	B 9495-3, B 9495-83
B 9499-3XM	Three Channels, up to 16-75 IPS MTU (3x16) PE Master Electronic Controller (MEC) (Also with NRZ Control Modification for one channel)	B 9495-2, B 9495-82
B 9499-3XH	Three Channels, up to 16-125 IPS MTU (3x16) PE Master Electronic Controller (MEC) (Also with NRZ Control Modification for one channel)	B 9495-3, B 9495-83
B 9499-4XM	Four Channels, up to 16-75 IPS MTU (4x16) PE Master Electronic Controller (MEC) (Also with NRZ Control Modification for one or two channels)	B 9495-2, B 9495-82
B 9499-2XH	Four Channels, up to 16-75 IPS MTU (4x16) PE Master Electronic Controller (MEC) (Also with NRZ Control Modification for one or two channels)	B 9495-3, B 9495-83
B 9499-21	One Channel, up to 8 GCR/PE MTU (1x8) (One Cabinet)	B 9495-22, B 9495-23, B 9495-24, B 9495-32, B 9495-33

B 9499-22	Two Channels, up to 8 GCR/PE MTU (2x8) (Two Cabinet)	B 9495-22, B 9495-23, B 9495-24, B 9495-32, B 9495-33
B 9499-23	Three Channels, up to 8 GCR/PE MTU (3x8) (Three Cabinet)	B 9495-22, B 9495-23, B 9495-24, B 9495-32, B 9495-33
B 9499-24		B 9495-22, B 9495-23, B 9495-24, B 9495-32, B 9495-33
B 9499-42	Exchange, Increases B 9499-22 to control up to 16 GCR/PE MTU (2x16)	
B 9499-43	Exchange, Increases B 9499-23 to control up to 16 GCR/PE MTU (3x16)	
B 9499-44	Exchange, Increases B 9499-24 to control up to 16 GCR/PE MTU (4x16)	

Printer Products

Style Number	Description
B 9251	230 CPS Serial Printer
B 9249-30	300 LPM Chain Printer
B 9249-50	500 LPM (48ch Set) 375 LPM (64ch Set) Chain Printer
B 9249-31	270 (64ch Set) Chain Printer
B 9249-37	270 (64ch Set) Chain Printer
B 9246-3	320 LPM (48ch Set) 300 LPM (64ch Set)
B 9246-6D	650 LPM (48ch Set) 600 LPM (64ch Set) Band Printer with ODEC Interface
B 9246-6A	650 LPM (48ch Set) 600 LPM (64ch Set) Band Printer with 2A Interface
B 9246-6H	650 LPM (48ch Set) 600 LPM (64ch Set) Band Printer with HSSI Interface
B 9246-6S	650 LPM (48ch Set) 600 LPM (64ch Set) Band Pointer with Centronics Interface
B 9246-12	1250 LPM (64ch Set) Band Printer with HSSI Interface
B 9246-13	1250 LPM (64ch Set) Band Printer with ODEC Interface
B 9246-20	2000 LPM (48ch Set) 1630 LPM (64ch Set) Train Printer with 2A Interface
B 9246-21	2000 LPM (48ch Set) 1630 LPM (64ch Set) Train Printer with HSSI Interface

EVALUATED SOFTWARE COMPONENTS

This appendix contains two lists. The first identifies the software items that are included in the TCB of UNISYS A Series. The second identifies UNISYS-supplied software items explicitly excluded from the evaluated configuration.

This Appendix does not explicitly list the benign software products which may be installed in A Series. Such a list would include numerous UNISYS products with the same restrictions: they were written in unprivileged languages, they are installed without privileged, and they are not system Libraries. Consequently, sites need only to confirm that the software they are about to introduce to their systems has been, or will be, compiled using such compilers.

TCB Software

The following list identifies TCB software that is included in the evaluated configuration. The list is composed of three parts: required trusted software, compilers, and optional trusted software. Required trusted software must be present in the evaluated configuration. The compilers identified below were examined during the evaluation documented by this report. Any or all of these compilers may be used in a C2 system if controlled as described in the Security Administration Guide. Optional trusted software is defined as software which, if include in a system, would be part of the TCB. This software has been examined and is considered part of the evaluated configuration.

Required Trusted Software

Master Control Program/Advanced Systems
- SYSTEM/ASD/AMLIP/MCP
- SYSTEM/ASD/A12A15/MCP
- SYSTEM/CONTROLLER (Originally bound into MCP/AS)
- SYSTEM/JOBFORMATTER (Originally bound into MCP/AS)

InfoGuard
- SYSTEM/IGSDASUPPORT
- SYSTEM/INFOGUARDSUPPORT

Command AND Edit Language
- SYSTEM/CANDE

Menu Assisted Resource Control
- SYSTEM/COMS/KERNEL
- SYSTEM/MARC/AGENDA/TDXXX
- SYSTEM/MARC/COMMANDER

Data Communication System
- SYSTEM/DATACOMINFO
- SYSTEM/DATACOMSUPPORT

System Software Utilities
- SYSTEM/CONFIGURATOR
- SYSTEM/GENERALSUPPORT
- SYSTEM/HELP
- SYSTEM/MAKEUSER
- SYSTEM/PLISUPPORT
- SYSTEM/PRINT/ROUTER
- SYSTEM/PRINT/SUPPORT
- SYSTEM/SIMPLEINSTALL
- SYSTEM/TRAINTABLES

Compilers

ALGOL Compiler
- SYSTEM/ALGOL

Cobol 74 Compiler
- SYSTEM/BDMSCOBOL74
- SYSTEM/COBOL74
- SYSTEM/MAXBDMSCOBOL74

DCALGOL Compiler
- SYSTEM/DCALGOL

DMALGOL Compiler
- SYSTEM/BDMSALGOL
- SYSTEM/DMALGOL

Fortran 77 Compiler
- SYSTEM/FORTRAN77
- SYSTEM/FORTRANSUPPORT

Multilingual System
- SYSTEM/MSGTRANS

Network Definition Language II Compiler
- SYMBOL/SOURCENDLII
- SYSTEM/NDLII

NEWP Compiler
SYSTEM/NEWP

PASCAL Compiler
SYSTEM/PASCAL
SYSTEM/PASCALSUPPORT

Program Binder
SYSTEM/BINDER

Report Program Generator II Compiler
SYSTEM/BDMSRPG
SYSTEM/RPG
SYSTEM/RPGSUPPORT

Sort Compiler
SYSTEM/SORT

Work Flow Language Compiler
SYSTEM/WFL (Originally bound into MCP/AS)

Optional Trusted Software

COMS

Communication Management System - Entry
SYSTEM/COMS/ENTRY

Communication Management System - Total
SYSTEM/COMS
SYSTEM/COMS/STATISTICS

Debuggers

ALGOL Test and Debug System
SYSTEM/TADSSUPPORT

Cobol 74 Test and Debug System
SYSTEM/C74TADSSUPPORT

Fortran 77 Test And Debug System
SYSTEM/F77TADSSUPPORT

Data Base Software

The use of any of the next four software products also requires the use of all software listed under DMSII below.

Advanced Data Dictionary System
- SYSTEM/ADDS/MANAGER

Data Base Analyzer
- SYSTEM/DBANALYZER

Data Base Certification
- SYSTEM/DBCERTIFICATION

InfoExec Environment
- SYSTEM/FORMS/ARCHIVEMANAGER
- SYSTEM/SCODE
- SYSTEM/SIM/DRIVER
- SYSTEM/SIM/MAPPER
- SYSTEM/SIM/UTILITY

DMSII
- SYSTEM/ACCESSROUTINES
- SYSTEM/DASDL
- SYSTEM/DMCONTROL
- SYSTEM/DMDATARECOVERY
- SYSTEM/DMDUMPDIR/LIBRARY
- SYSTEM/DMINTERFACE
- SYSTEM/DMRECONFILTER
- SYSTEM/DMRECOVERY
- SYSTEM/DMUTILITY

Miscellaneous

ReprintS
- SYSTEM/PRINT/REMOTE/SERVER

Interactive Datacomm Configurator
- SYSTEM/IDC
- SYSTEM/IDC/AGENDA/TDXXX

Screen Design Facility
- SYSTEM/SCREENDESIGN/MANAGER

Image Printer Support Facility
- B929030/DIAGNOSTICS/VSID/LSDIAG
- B929030/SOFTWARE/VSID/020001
- DEFAULT/FORM
- DEFAULT/PCF
- FONT/IP240A/DEFAULT
- SYSTEM/IPSUPPORT

System Software Utilities
- SYSTEM/HARDCOPY
- SYSTEM/IMG/AGENDA/TDXXX
- SYSTEM/KEYEDIO

Excluded Software

The software identified below is non-evaluated trusted software. This software must not be used in a C2 operating environment.

APL/700
SYSTEM/APL

APLB Language
SYSTEM/APLBSUPPORT

Basic
SYSTEM/BASIC

Fortran IV Level H Compiler
SYSTEM/FORTRAN

Master Control Program/Advanced Systems
SYSTEM/ASD/MCP/DIAGNOSTICS

Transaction Processing System
SYSTEM/TFL

Cobol 68 Compiler
SYSTEM/BDMSCOBOL
SYSTEM/COBOL

PL/I Compiler
SYSTEM/BDMS/PL/I
SYSTEM/PL/I

BNA Host Services
SYSTEM/HOSTSERVICES
SYSTEM/HOSTSERVICES/DIAGNOSTICS

BNA Network Services
SYSTEM/BNAV1/NETWORKSERVICES
SYSTEM/BNAV1/NETWORKSERVICES/DIAGNOSTICS
SYSTEM/ICP/DUMP/LIB
SYSTEM/BNAV2/ENVIRONMENT
SYSTEM/BNAV2/ENVIRONMENT/DIAGNOSTICS
SYSTEM/BNAV2/NPSUPPORT
SYSTEM/BNAV2/NPSUPPORT/DIAGNOSTICS
SYSTEM/ICP1/FIRMWARE

GEMCOS
- SYSTEM/GEMCOS/MCS/ADVANCED
- SYSTEM/GEMCOS/LIB
- SYSTEM/GEMCOS/MCS/BASIC
- SYSTEM/GEMCOS/MCS/MASTER
- SYSTEM/GEMCOS/MCS/TOTAL

HOSTNSP
- SYSTEM/HOSTNSPSUPPORT

X.25 MCS
- SYSTEM/X25
- X25/APPLICATION

Command AND Edit Language
- SYSTEM/CANDE/DIAGNOSTICS

Diagnostic Message Control System
- SYSTEM/DIAGNOSTICSMCS

Remote Job Entry
- SYSTEM/RJE
- SYSTEM/RJE/DIAGNOSTICS

ReprintS
- SYSTEM/PRINT/BNAROUTER

Master Control Program
- SYSTEM/AMLIP/MCP
- SYSTEM/MCP
- SYSTEM/MCP/DIAGNOSTICS

System Software Utilities
- SYSTEM/BASICSUPPORT
- SYSTEM/IADMAPPER
- SYSTEM/OMRSUPPORT

GLOSSARY

ACCESSCODE

An identification code subordinate to a USERCODE that can be specified in the USERDATAFILE as required along with a USERCODE/PASSWORD combination (and, sometimes, with an associated password of its own). An ACCESSCODE is used to further establish a user's identity, control security, and allow access to disk files.

Address Couple

The form of an address. A pair of integers (lambda, delta) that respectively designate the position of a procedure in the lexicographic nesting structure of the program and the position of a word in the activation record of that procedure.

APASSWORD

A password which is associated with an ACCESSCODE.

ASD

Actual Segment Descriptor. An ASD is a three-word structure containing the absolute memory address of the first word of an actual segment, status flags and the length of the segment. MCP/AS maintains actual segment descriptors in the ASD table.

BNA

Burroughs Network Architecture. The network architecture used on Burroughs computer systems to connect multiple, independent Burroughs computer systems into a network for distributed processing and resource sharing. BNA software is not included in an evaluated configuration.

BOSR

Base of Stack Register. This hardware register identifies the base of the user's stack space in an active stack.

CANDE

Command and Edit. A UNISYS Message Control System (MCS) that provides generalized file preparation and updating capabilities and task control in an interactive, terminal-oriented environment.

CHARGECODE

An account number of the account to which the cost of a computer session is to be charged.

COMS

Communication Management System. A general Message Control System (MCS) that supports a network of users and provides them with a consistent, on-line interface between the Data Communication subsystem and application programs that process transactions associated with remote terminals.

Controlled

A possible value of the SECURITYTYPE file attribute. When a file has a SECURITYTYPE of controlled, the file's Guardfile is examined prior to granting access to any user, including the owner.

CSD

Code Segment Descriptor. A pointer to a virtual segment containing executable code.

DCALGOL

Data Communications ALGOL. A language based on UNISYS extended ALGOL that contains extensions allowing it to be used to write Message Control Systems (MCSs) and other specialized system programs.

DD

Data Segment Descriptor. A pointer to a virtual segment containing data.

Dialog

An instance of communication with a COMS window.

DLP

Data Link Processor. An element of the I/O subsystem: it interfaces between a peripheral unit or controller and the I/O processor of A Series system. DLPs are specialized to the type of unit.

DMALGOL

Data Management ALGOL. A superset of DCALGOL with extensions. DCALGOL is not for users' applications and is not supported outside the DMSII environment.

DMSII

Data Management System II. A specialized system software package used to describe a data base and maintain the relationships between the data elements in the data base.

DoD

Department of Defense.

EPL

Evaluated Products List. A list of products that have been subjected to formal product evaluation by the National Computer Security Center, and their assigned ratings.

Family

One or more disks or tapes logically grouped together and treated as a single entity by the system.

FAST

File Access Structure. A system data structure used to locate files on disk.

FIB

File Information Block. The FIB contains, directly or indirectly, all of the elements of a file's structure.

FIBOPEN

The MCP/AS entrypoint responsible for opening all kinds of files. For disk files, spooled files, and port files, FIBOPEN directly enforces access rights.

File

A collection of data considered a unit.

FILE

The programmatic object declared in a program to provide a conduit for information transfer.

Filekind

The attribute of a disk file that specifies its nature and purpose; MCP/AS applies special protection to files of certain Filekinds.

FLAT

An MCP/AS table associated with each disk family that contains information describing the attributes, location and name of files in that family.

Guarded

A possible value of the SECURITYTYPE file attribute. The owner of a file that is guarded is allowed access to the file regardless of the access list in the guardfile, while the access of all other users is determined by the guardfile.

HDU

Host Dependent Unit. The I/O processor on some A Series systems (e.g., A12, A15).

History Link

A self-relative index in an MSCW, designating the prior MSCW in the same stack.

InfoGuard

The UNISYS security-enhancement software for large systems. InfoGuard provides such features as password management, selective logging and auditing, tape volume security, and simplified system security configuration.

IOCB

Input/Output Control Block. A data structure used for communication between the host system and the MLIP or HDU.

IRW

Indirect Reference Word. An IRW is a reference to a word within the same or another stack. An IRW has two forms: NIRW and SIRW.

LEB

Label Equation Block. A representation of a file's attributes that are declared at run-time.

Lexicographic Level

The nesting depth of a procedure declaration in the static structure of a program.

Library

A collection of one or more named routines or "entry points" that are stored in a file and can be called by other programs.

LOSR

Limit of Stack Register. This hardware register identifies the limit of the user's stack space in an active stack.

MARC

Menu-Assisted Resource Control. A menu-driven interface and transaction processor for users and operators of UNISYS A Series systems.

MCP/AS

Master Control Program/Advanced System. The operating system on UNISYS A Series systems. MCP/AS controls the operational environment of the system by performing disk management, memory management, interrupt processing, process control, I/O operations and other system overhead functions.

MCS

Message Control System. A program that controls the flow of messages between terminals, application programs and the MCP/AS. UNISYS A Series MCSs in an evaluated configuration are CANDE and Communication Management System (COMS).

Mix Number

The number by which a task or job is referenced while it is executing. This number also appears in every audit record.

MLIP

Message Level Interface Processor. The I/O processor associated with a central processing unit on some A Series systems (e.g., A3, A5, A9).

MSCW

Mark Stack Control Word. The first word in the stack linkage at the base of an activation record. An MSCW holds a history link, environment link, lexicographic level, and inactive/entered bit. Only the history link is valid in an inactive MSCW.

Multidrop

A data communication subsystem capable of controlling two or more connection endpoints on the same physical line. This method of connecting terminal is not acceptable in a C2 configuration.

NCSC

National Computer Security Center.

NDL II

Network Definition Language II. The UNISYS language used to physically, logically, and functionally describe the datacomm subsystem on Message-Level Interface Processor (MLIP) and Host Dependent Unit systems.

NEWP

NEWP is an ALGOL-based, high-level language designed for implementation of system software. NEWP includes low-level constructs for I/O, memory and processor management. It is the programming language used by UNISYS to write the operating system (MCP/AS). Code files which are produced by the NEWP compiler are executable, except for unsafe code files, which are non-executable.

NIRW

A reference to a word within the direct addressing environment of a procedure. An NIRW contains only an address couple.

ODT

Operator Display Terminal. The system console device that allows the operator to enter commands directly to the operating system to perform various functions, and to see information displayed by the system.

PAST

Pack Access Structure. A system data structure used to locate FAST entries for a family.

PCW

Program Control Word. The defining reference to a procedure. A PCW holds a code-stream pointer and certain other initial state values for the procedure. The activation record in which the PCW appears defines the global addressing environment for the procedure. A PCW may also define the destination for a dynamic branch operator, in which case only the code stream pointer is significant.

PIB

Process Information Block. The PIB contains task attribute information.

Port FILE

A conduit for passing messages between two or more concurrently running tasks.

RCW

Return Control Word. The second word in the stack linkage at the base of an activation record. The RCW contains a code stream pointer and certain other state information to permit return to the code stream following a procedure invocation.

Remote FILE

A conduit between a program and one or more terminals.

SAG

Security Administration Guide. The UNISYS A Series version of a Trusted Facility Manual. See TFM.

SDA

System Data Access.

SECADMIN

1) Security Administrator. The person responsible for establishing and maintaining system security. 2) System option specifying a mode of operation where some set of users will be designated as Security Administrator. 3) User option identifying a user as a Security Administrator.

SFOPG

Security Features Operation and Programming Guide. The UNISYS A Series version of a Security Features User's Guide. See SFUG.

SFUG

Security Features User's Guide. A single summary, chapter, or manual in user documentation that describes the protection mechanisms provided by the TCB, guidelines on their use, and how they interact with one another.

SIRW

A context-independent reference to a word in an activation record. An SIRW is generated from an NIRW by replacing the lexicographic level of the NIRW with a stack number and displacement to the appropriate activation record.

SPO

Supervisory Printer Output. An obsolete term for the Operator Display Terminal. See ODT.

Station

A data terminal.

SUMLOG

A comprehensive event-log file created and maintained by the system that contains information on resource usage, hardware errors, security violations, system messages, and job file activity.

Task

A task is the UNISYS term for a process.

TCB

Trusted Computing Base. The totality of protection mechanisms within a computer system - including hardware, firmware, and software - the combination of which is responsible for enforcing a security policy.

TFM

Trusted Facility Manual. A manual addressed to a system administrator that presents cautions about functions and privileges that should be controlled when running a secure facility.

TSOW

Top Stack Control Word. A word at the base of a process stack. The TSOW contains fields that preserve the settings of the 'F' and 'S' registers while a stack is inactive.

USERCODE

An identification code used to establish user identity, control security and provide for segregation of files. USERCODEs can be applied to every task, job, session, and file on the system.

USERDATAFILE

A system data base that defines valid USERCODEs and contains various data about each user (such as ACCESSCODEs, passwords, and CHARGECODEs) and the population of users for a particular installation.

WFL

Work Flow Language. The UNISYS A Series language used to write jobs that control the flow of programs and tasks on the operating system (batch).

Window

A communication avenue via COMS to another MCS, a transaction processor, or a remote FILE.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS NONE
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT DISTRIBUTION UNLIMITED
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) CSC-EPL-87/003	5. MONITORING ORGANIZATION REPORT NUMBER(S) S228,515
6a. NAME OF PERFORMING ORGANIZATION National Computer Security Center 6b. OFFICE SYMBOL (If applicable) C12	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code) 9800 Savage Road Ft. George G. Meade, MD 20755-6000	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 NUMBERS PROGRAM ELEMENT NO / PROJECT NO / TASK NO. / WORK UNIT ACCESSION NO

11. TITLE (Include Security Classification)

(U) Final Evaluation Report, UNISYS Corporation A Series MCP/AS Release 3.7

12. PERSONAL AUTHOR(S)
Cornelius Haley, Mary Schanken, Sheryl Luera*, Jerzy Rub*, et al (* Aerospace Corp.)

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final	FROM _____ TO _____	870805	104

16. SUPPLEMENTARY NOTATION

17 COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	A Series MCP/AS InfoGuard Security Enhancements C2 Trusted Computer System Evaluation Criteria NCSC EPL

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

The security protection provided by UNISYS Corporation's A Series MCP/AS Release 3.7 was evaluated by the National Computer Security Center (NCSC). The NCSC evaluation team has determined that the highest class at which the A Series satisfies all the specified requirements of the Trusted Computer System Evaluation Criteria, dated December 1985, is class C2, and therefore has been assigned a class C2 rating by the NCSC.

A system that has been evaluated as being a class C2 system provides a Trusted Computing Base (TCB) that enforces discretionary access control and, through the use of audit capabilities, accountability for the actions users initiate.

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 (Include Area Code) (301)859-4458 22c. OFFICE SYMBOL C/C12

DD Form 1473, JUN 86 *Previous editions are obsolete* SECURITY CLASSIFICATION OF THIS PAGE