NCSC-EPL-86/007

(2)

NATIONAL COMPUTER SECURITY CENTER

FINAL EVALUATION REPORT OF GOULD, INC. COMPUTER SYSTEMS DIVISION UTX/32S

Release 1.0

31 December 1986

Distribution
Inside Cover

FINAL EVALUATION REPORT

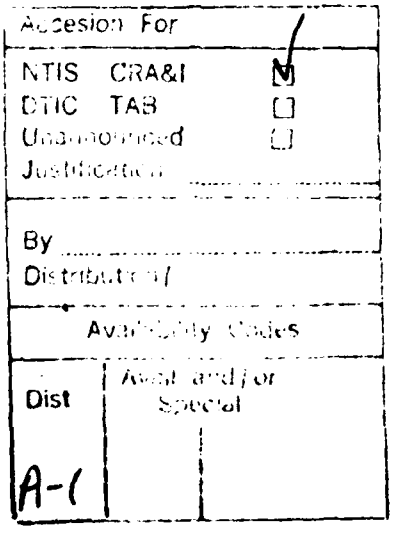
GOULD, INC., COMPUTER SYSTEMS DIVISION

UTX/32S RELEASE 1.0

NATIONAL
COMPUTER SECURITY CENTER

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

December 31, 1986



CSC-EPL-86/007
Library No. S228,373

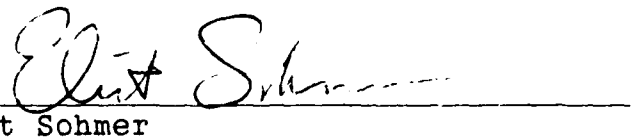
This page intentionally left blank.

CSC-EPL-86/007
Library No. S228,373

FOREWORD

This publication, the Final Evaluation Report Gould, Inc., Computer Systems Division UTX/32S Release 1.0, 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 Gould's UTX/32S Release 1.0 operating system. The requirements stated in this report are taken from Department of Defense Trusted Computer System Evaluation Criteria dated December 1985.

Approved:



December 31, 1986

Eliot Sohmer
Chief, Product Evaluations
and Technical Guidelines
National Computer Security Center

ACKNOWLEDGEMENTS

Team Members

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

Cornelius J. Haley
Steven M. LaFountain
Holly M. Traxler

National Computer Security Center
Ft. George G. Meade, MD 20755-6000

Dixie B. Baker, Ph.D.
R. Leonard Brown, Ph.D.
Kenneth B. Elliott III

The Aerospace Corporation
El Segundo, CA 90245

CONTENTS

This page intentionally left blank.

EXECUTIVE SUMMARY

The security protection provided by Release 1.0 of the Gould UTX/32S(1) operating system, configured according to the most secure manner described in the Trusted Facility Manual (see page 32, "Evaluation as a C2 System, Trusted Facility Manual"), running on a Gould PowerNode(2) processor has been evaluated by the National Computer Security Center (NCSC). The security features of UTX/32S were evaluated against the requirements specified by the DoD Trusted Computer System Evaluation Criteria (the Criteria) dated December 1985.

The NCSC evaluation team has determined that the highest class at which UTX/32S satisfies all the specified requirements of the Criteria is class C2, and therefore, using the specified hardware (see page A-1, "Evaluated Hardware Components"), Release 1.0 of UTX/32S (see page B-1, "Evaluated Software Components"), configured in the most secure manner described in the Trusted Facility Manual, has been assigned a class C2 rating.

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

The UTX/32S system consists of the UTX/32S operating system running on a Gould PowerNode minicomputer. The largest UTX/32S system utilizes two tightly coupled processors and can support up to 128 users.

The UTX/32S operating system is a general purpose, time-sharing system that provides two basic security mechanisms. The restricted environment provides separation between administrative and normal users and files. The protection bit mechanism provides discretionary access control.

UTX/32S differs from standard UNIX in that the restricted environment mechanism is included, the setuid and setgid mechanisms have been removed, and trusted servers have been implemented to provide the capabilities formerly provided by the setuid and setgid mechanisms.

---

(1) UTX/32S is a trademark of Gould, Inc.

(2) PowerNode is a trademark of Gould, Inc.

This page intentionally left blank.

INTRODUCTION

In May 1986, the National Computer Security Center (NCSC) began a formal product evaluation of Release 1.0 of UTX/32S(1), a Gould, Computer Systems Division product. The objective of this evaluation was to rate the UTX/32S system against the Criteria, and to place it on the Evaluated Products List (EPL) with a rating. This report documents the results of that evaluation. This evaluation applies to UTX/32S Release 1.0 available from Gould in May 1986.

Material for this report was gathered by the NCSC UTX/32S evaluation team, through documentation, interaction with system developers, and experience gained while using a UTX/32S system.

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, as a result of National Security Decision Directive 145 (NSDD-145), 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

---

(1) UTX/32S is a trademark of Gould, Inc.

manufacturer's design for either a new trusted product or for security enhancements to an existing product. Since the developmental phase is based on design documentation and information supplied by the industry source, it involves no "hands on" use of the system. The developmental phase results in the production of an Initial Product Assessment Report (IPAR). The IPAR documents the evaluation team's understanding of the system based on the information presented by the vendor. Because the IPAR contains proprietary information, distribution is restricted to the vendor and the NCSC.

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

Document Organization

This report consists of four major sections and three appendices. Section 1 is this introduction. Section 2 provides an overview of the system hardware and software architecture. Section 3 provides a mapping between the requirements specified in the Criteria and the UTX/32S features that fulfill those requirements. Section 4 presents comments from the evaluation team on specific UTX/32S features. The first two appendices identify the specific hardware and software components to which the evaluation applies. The third appendix, Appendix C, contains a list of references used by the evaluation team.

SYSTEM OVERVIEW

This section provides a summary of the UTX/32S system and its protection mechanisms. Although most security mechanisms are covered in this section, greater detail can be found in section 3, "Evaluation as a C2 System."

UTX/32S is a UNIX-based(1) operating system that incorporates features from both 4.2 BSD and System V UNIX systems. A number of features have been added to enhance the security of UNIX, and others have been removed for the same reason. The following section details these changes, as well as touching on more general issues concerning UTX/32S.

Hardware

The UTX/32S system runs on Gould PowerNode 6000 and 9000 series processors. The PowerNode can be configured with either a single processor, or a CPU/IPU(2) (Internal Processing Unit) pair. These processors are identical and interchangeable -- the processor operating as the CPU does normal processing as well as processing system calls, interrupts, and traps. The IPU does not process system calls, and recognizes a very limited subset of the interrupts and traps. A minimum of 3 Megabytes (MB) of memory is needed to run UTX/32S.

The CPU is capable of running in either privileged or unprivileged mode, depending on the status of bit 0 of the PSW. The UTX/32S kernel runs in the privileged state, although the kernel is only a part of the TCB (i.e., the entire TCB does not run in the privileged state.) Bit 0 is used by the CPU to perform all interrupt related instructions, such as enable interrupt or request interrupt; all instructions that can modify the memory mapping registers; all I/O instructions; and all instructions that can place the machine in a state that requires operator intervention to continue processing, such as HALT. Actions which cause the processor to change from the unprivileged state to the privileged state include:

---

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

(2) IPU is a registered trademark of Gould, Inc.

- An interrupt from an external event or the I/O system.

- A hardware trap caused by addressing nonpresent (virtual) memory, executing an undefined instruction, the execution of a privileged instruction by an unprivileged program, or writing to protected memory.

- A hardware trap caused by a nonrecoverable condition such as an uncorrectable error on a memory read, or an arithmetic exception.

- The execution of the supervisor call instruction by a user requesting monitor services.

- A system reset.

Interrupts and traps in the system are serviced mainly by the CPU. Interrupts are prioritized, scheduled, and sometimes deferrable (in the case of a higher priority interrupt pending). They are asynchronous, and there is no queue for waiting interrupts; this means a device must continue to assert its interrupt line until it is recognized by the CPU. I/O interrupts are non-interruptable -- a hardware capability blocks all other I/O interrupts until the previous I/O completes. In addition, all interrupts operate in the "mapped" environment, discussed below. In contrast, traps are non-prioritized (with the exception of the power fail trap, which supersedes all other traps, and all interrupts) and non-deferrable. However, a trap which is received by the IPU which requires the use of the CPU will defer until the CPU can service it.

The Input/Output Interface (IOI) is a part of the kernel. The IOI provides an interface between the device drivers and the devices themselves. This interface is shared by all of the device drivers. The IOI provides code in a single place common to more than one device driver, thus simplifying each device driver.

In addition, the IOI directs the initialization of the I/O system during system initialization and provides privileged user processes with a maintenance interface to devices. The IOI has simplified the design of the individual drivers by moving certain error recovery operations (such as issuing a sense status channel program) into a separate section of code. This reduces the number of states required by each driver for error recovery and a reasonable processing of situations such as getting a unit check during a sense status operation. The IOI allows drivers to specify whether they need address mapping or error recovery assistance from the IOI.

There are a number of addressing modes available on the hardware: base register mode and nonbase register mode (indicated by bit 6 of the PSW), and mapped and unmapped environments (indicated by bit 31 of the second PSW). Nonbase register mode uses an index register, an indirect bit, and bits 12-31 of a general purpose register to indicate an address. This mode will not be discussed, as UTX/32S does not use it. The unmapped environment uses logical addresses as physical addresses (i.e., no address translation takes place). There is no memory protection in the unmapped environment. A process uses the unmapped environment only when doing physical memory copy or zero operations. Since these operations take place in the kernel, security is preserved.

UTX/32S primarily uses the base register, mapped environment for addressing, which will now be discussed. The base registers used are referred to as B0-B7, and contain addresses of portions of the process' address space (see page 8, "System Overview, Subjects"). The mapped environment uses memory allocation and protection (MAP) blocks of 2K length that correspond to physical memory. A MAP image descriptor list (MIDL) is kept to point to the MAP blocks. For UTX/32S, the MIDL is referred to as the page table, and the actual MAP blocks are the page table entries. The page tables for all processes reside in the kernel, and are linked to a process via the proc structure for each process. Having the page tables reside in the kernel space eliminates the need to swap page tables on context switch. The list of page table entries is augmented with an array of 32-bit words that hold disk addresses corresponding to the memory pages of an uninitialized or paged-out page. The address translation process is as follows.

A logical address is 24 bits long. It is generated by the summation of a base register (bits 13-15 of the opcode), a quantity in a general purpose index register (bits 9-11 of the opcode), and a 16 bit offset directly from the opcode. The first 11 bits of the logical address are an index into the page table, (indicating a certain page table entry), which provides the upper 11 bits of the physical address (these two 11 bit quantities are distinct). The lower 13 bits of the logical address provide the lower 13 bits of the physical address (these two 13 bit quantities are identical). If the page table entry indicates a paged out condition on the page to which it refers, a page fault occurs.

The Filestore

Manipulation of files is the heart of any UNIX-like system, and UTX/32S is no exception. There are three basic types of files: storage files, device files (sometimes called device special files), and directory files. Storage files contain data or executable images, and correspond to the traditional concept of "file." Directory files contain a list of inode numbers and the corresponding names of all files in that directory. The inode for device special files contains the class of device (such as "block" or "character") and a "device number" (composed of a major and minor device number). The device driver itself only interfaces with the rest of the kernel by 1) the entry point listed in its class, as indexed by the major device number, 2) using the common buffering system, and 3) using the common low-level hardware support routines and data structures (the IOI).

The term "filestore" is distinct from the concept of the "file system," as the filestore may comprise several file systems. Each file system may be mounted on a physical device (e.g., a disk drive) or a logical device (e.g., a partition on a disk drive); however, there may only be one file system mounted per logical device, and a file system cannot span physical hardware boundaries. References to files are resolved by parsing a "pathname." A pathname for a file consists of a string of characters, starting with root ("/"), and continuing with a hierarchical list of directories, ending in the name of the file (e.g., /f/alpha/beta refers to the file "beta" in the sub-directory "alpha," in the sub-directory "f," in root). If, in the resolution of the path, a mount point is found (a mount point is where a file system is grafted onto the filestore, this information appears in the inode), a mount table resident in main memory supplies the needed device information.

A link is an entry which appears in a directory, and which refers to a file (or possibly another directory). There are two types of links which may be made to these files: hard links and symbolic links. A hard link is indistinguishable from the original directory entry; any changes to a file are effective independent of the name used to reference the file. Hard links may not span file systems and may not refer to directories. Technically, all files have at least one hard link. A symbolic link contains the name of the file to which it is linked. The referenced file is used when an open operation is performed on the link, and a stat on a symbolic link will return the linked-to file. Symbolic links may span file systems and may refer to directories.

Although a file or directory need not have a unique name, a single pathname will resolve to the same file (inode). The only case where (externally) it may seem as if the pathname was resolved to two different files was if a chroot had been issued. However, chroot only changes that process' notion of root; internally the pathname is expanded (the path of the root directory is prepended to the pathname entered by the user) so that the pathname resolves to a unique file.

The Restricted Environment

The domain separation mechanism in UTX/32S is the restricted environment (RE) (see Figure 1). The chroot system call divides the UTX/32S filestore into two sections: the RE and the administrative environment (AE). The RE is a subtree in the overall filestore, the root of which is the directory /renv/renv1 of the AE. Any process that runs inside the restricted environment (i.e., has its home directory in that file system) cannot access any data or executable outside of the RE, except through a secure socket (see page 10, "System Overview, Secure Sockets and Trusted Servers"). Even then, the execution is performed on the process' behalf by a trusted server. A process inside the RE is also severely limited as to what privileges it can obtain -- for instance, a process inside the RE can not become a "superuser" process, a term which in UNIX indicates a process that may access any file in the filestore, regardless of discretionary access controls which may restrict this access. Only the processes of administrators (users who can log into the system outside of the Restricted Environment) have the ability to become "superuser" processes.

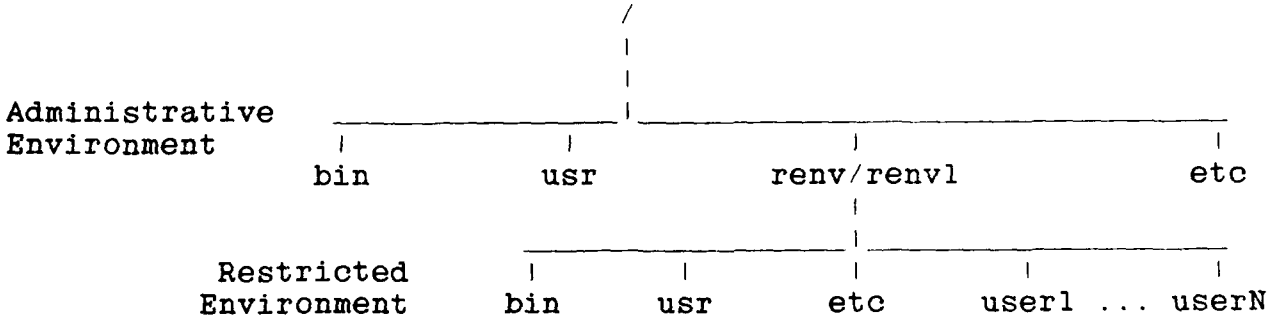


Figure 1. The Restricted Environment

The data and executables which lie inside the AE are considered part of the Trusted Computing Base (TCB). All administrators of the system have home directories which are inside the AE, hence the processes that act on their behalf are also part of the TCB. All device control files and trusted servers reside inside the AE.

Subjects

Subjects in UTX/32S are the processes which act on behalf of the users of the system. Each process is allocated space in the virtual memory of the system. The lower 2MB of each process is mapped directly to the kernel to avoid re-mapping on a context switch, while the upper regions consist of the user stack, static storage (which is base-register addressable), and non-initialized data, called BSS. The process operates in a demand paged environment, with zero fill being performed on page-in. The base registers (B0-B7) are assigned as follows:

- B0 - used only occasionally as a scratch register when data is moved between two memory locations.

- B1 - points to the nonbase-addressable entry point of a routine while that routine is active. It is used for branches within that routine.

- B2 - is the stack pointer. The stack grows from high address to low.

- B3-B7 - assigned values by the link editor (ld) when the program is linked. The link editor first ensures all base-addressable objects can be accessed through one of the base registers. It then fills in the base register and offset fields of relocated instructions accordingly.

All processes in UTX/32S are created in a similar manner. The parent process performs a fork system call, which makes a copy of the parent process; the only differences being the process ID and the base register values. The new process, called the child, then performs an exec system call, and passes the desired executable image name. exec copies the executable image into the new process' text space (or copies a pointer, if the text is sharable), and passes control to the designated entry point.

There are two types of subjects in UTX/32S -- privileged and unprivileged, depending on the values of its IDs. Each subject has three IDs associated with it -- a User ID, an Environment ID,

and a Logging ID. Each account has a unique logging ID which is used to keep track of audit information. The environment ID is used to determine whether or not the process resides inside the RE. A user ID is generally non-zero. There are two exceptions to this: 1) All system processes run with User ID = Environment ID = Logging ID = 0 2) When an administrative user (i.e., one who is not in the RE) becomes the superuser (by requesting a superuser shell via the sush or sucsh commands), the user ID of that process is set to zero. Logging is still done, however, with that administrator's logging ID. A process in the RE can never have a zero environment ID or a zero user ID. There is another ID used for Discretionary Access Control (DAC) purposes called the group ID. A set of groups is associated with each user in the /etc/groups file. Each group has a distinct ID number, and these numbers are used in determining access to a given file (see page 13, "Evaluation as a C2 System, Discretionary Access Control").

In standard UNIX, the parent has the right to specify environment variables for the child when forking. UTX/32S limits the set of environment variables a child can have to a subset of those of the parent.

Objects and Object Protection

Objects in UTX/32S consist of files (including directories), and I/O devices. There are several types of files. Traditional storage files contain data and executable images. Directory files contain directory information. Device files allow a process to access a device by writing to or reading from a file.

Devices are divided into three categories: (1) "privileged" which can be accessed only by a superuser process (e.g., disks); (2) "allocatable" which can be accessed only by the owner or by a superuser process (e.g., magnetic tape drives, communication lines); and (3) "discretionary" which can be accessed by any user (e.g., /dev/null). Further, the Security Policy Model for Gould UTX/32S, Release 1.0 [13] considers I/O devices to be data objects and defines three categories: (1) login devices; (2) user devices, for which login is not allowed, but which may be allocated to a user (e.g., magnetic tape drives, external communication lines); and (3) system devices, which are directly accessible only by the TCB (e.g., line printers).

The traditional UNIX file protection bits form the discretionary access control (DAC) mechanism used in UTX/32S. These consist of nine bits associated with each file -- three for the owner

(creator), three for the group to which the file belongs, and three for the rest of the users on the system. The three bits represent read, write, and execute access, respectively. If a bit is set, the appropriate process (owner, object's group, other) has the respective access to the object. Initial DAC information associated with the object is governed by the umask system call, through which a user can define which protection bits are set when a file is created. UTX/32S ensures that, no matter what umask is set as, the world (defined as all other users on the system outside of the object's owner and group) has no access to a newly created object. For more information on the discretionary access control mechanism (see page 13, "Evaluation as a C2 System, Discretionary Access Control").

Some files are owned by the root; that is, the user ID of the owner is zero. These files are the system files, such as the /etc/passwd file, the /etc/rc file, and the spool files. Unprivileged users (users with a non-zero User ID) can not modify root-owned files and can not move or delete files resident in root-owned directories.

Secure Sockets and Trusted Servers

The elimination of setuid and setgid files from UTX/32S, along with the need to have communications between the RE and the AE (ps command, for example) caused Gould to introduce the secure socket mechanism. This is a variant of the BSD 4.2 socket mechanism, and provides for a secure communications link from processes of lower privilege to those of higher privilege.

A trusted server is a daemon that will perform a service that requires privileges that a user requesting the service does not possess. For instance, ps requires the use of /dev/kmem, which is kernel memory and is a privileged file. A trusted server is initiated by init from /etc/rc at system startup. The trusted server creates a secure socket, binds to that socket, and then listens to the socket for users requesting its services. Only a process with user ID = 0 can bind to a secure socket. The secure socket's domain is specified by the constant argument AF_SECURE to the socket system call.

A client process is a process that wishes to establish communications with the trusted server. This process can be in either the RE or the AE, but is usually in the RE. The process first specifies the secure socket domain, and then connects to the socket that the desired trusted server is listening to. This is transparent to the user issuing the request. When a client

wants to establish communications with the trusted server, a protocol control block (PCB) is set up by either the system call or the socket call. The PCB contains information about the user that will be used to ensure the client has the proper privileges to access the trusted server. If the PCB is not present, no connection will be made.

The trusted servers in UTX/32S are: the administrative server, the authorization server, the device control server, the generic server, the line printer server, and the mail server. The administrative server is the mechanism used to allow a user logged on as an administrator to set his user ID to 0. The authorization server maintains the password and group files. The device control server controls access to privileged and allocatable devices. The generic server is used as an interface for UNIX commands that require special privileges (e.g., ps is performed by the generic server). The line printer and mail servers control access to the line printer and mail systems.

When a client first establishes a communications channel (secure socket) with a trusted server, it is communicating with a "shared server." The shared server is the trusted server started by init. Once the shared server has validated the client (via an ioctl system call to obtain the PCB), it forks a copy of itself and severs its connection to the client. The shared server is then ready to accept requests from other clients. The child (forked) process, called the dedicated server, is then in communication with the client via a secure socket, and will perform its function on behalf of the client.

Another function of the secure socket mechanism is to regulate the flow of information (file descriptors) between the client and the server. A client socket can send file descriptors to a server socket but cannot receive file descriptors from a server socket. In turn, the server socket can accept file descriptors from the client socket but can not send file descriptors to the client socket.

Changes to Standard UNIX

Certain features that are standard to UNIX are the cause of security concerns. UTX/32S has addressed these issues, and has presented the following solutions.

- The setuid and setgid mechanisms allow a process to take on the user ID or group ID of the executable. Since there are many files in standard UNIX that use this mechanism, it is

hard to verify that access permissions to these files are set correctly. Gould has entirely eliminated the setuid and setgid bits from executable files. Services which formerly needed these bits to run are now implemented as trusted servers.

- Concerns over the corruptability of sensitive files and/or directories (e.g., mail) were solved by moving these sensitive files outside the RE, where the general user can not get access to them except through a trusted server.

- The special files /dev/mem and /dev/kmem are readable and writable in standard UNIX, if the access bits are set. In UTX/32S, these devices are readable only by a superuser process, and the ability for anyone to write to these devices has been removed.

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

Discretionary Access Control in UTX/32S is implemented using a protection bit mechanism. Each object in the storage system has associated with it twelve bits that signify the access modes with which users can access the object. Also associated with each object is the name of the owner of the file and the name of the group to which the file belongs.

The first two protection bits are the setuid and setgid bits. These bits are not used in UTX/32S. The UTX/32S kernel ignores these bits when a file is accessed and/or executed. The kernel also ensures that these bits can not be set on any file.

The third bit is the "sticky" bit. When this bit is set, the system will not destroy the swap image of the associated program. When a user finishes executing the program, its image is swapped out and saved until initiation by the next user. This saves time in that the system does not have to retrieve the program from the file system on every initiation. This bit can only be set by a system administrator.

The last nine bits can be set by a system administrator or by the owner of the object through the use of the chmod command. Of these nine bits, the first three signify read/write/execute access modes respectively for the object's owner. If the bits are set then the corresponding access modes are allowed. If the bits are not set, the corresponding access modes are denied. The second (middle) three bits specify the access modes for other users who are members of the group to which the object belongs. The last three bits specify the access allowed to all other users (i.e., the public access permission to the object).

The UTX/32S DAC mechanism can be used to provide access control to the granularity of a single user. This requires that a system administrator set up the group structure such that all combinations of users exist. This is not a very convenient implementation of single user granularity. However, this problem is common to all protection bit based access control mechanisms.

Upon creation, all objects are given an initial (default) protection bit setting. This setting is determined through system-wide and user-specific umasks. In UTX/32S, the system-wide umask allows read/write access for the owner of the object and for the group in which the object is being created, and null public access by default. After the system-wide umask is applied to the newly created object, the user-specific umask is then logically ANDed with the protection bits. This allows users to further restrict but not expand the initial access permissions to objects that they create. Users can, however, specify any combination of access after the object is created by using the chmod command. The system administrator is responsible for initially setting the user-specific umask for each user when the user is installed on the system. To provide default protection that initially limits access to only the owner, a system administrator must set the user-specific umask such that group access is disallowed. This procedure is described in the Trusted Facility Manual.

After a user is installed in the system, the user may change his own user-specific umask to allow or deny access permission to himself and the group in which the file belongs. Also, after object creation, the owner may set the protection bits to allow or deny any type access by explicitly specifying that access using the chmod command.

The owner of an object may be changed only by a system administrator. There are no restrictions on who the new owner may be.

The group associated with an object may be changed by the system administrator or by the owner of the object. However, if the owner is to change the group, the owner must be a member of the group to which the change is being made. If the owner is not a member of the new group, the change will not occur.

For files, the meanings associated with the access modes are clear. Read access allows the user to read the contents of the file. Write access allows a user to write to (change or append to) the file. Execute access allows the user to execute the file as a program.

For directories the meanings of the access modes are not as obvious. For directories, read access allows a user to list the names of the objects contained in the directory. Write access allows a user to create objects in and delete objects from the directory. Execute access allows a user to use the directory in a pathname (i.e., it allows a user to search through the directory without actually being able to see the names of the objects in the directory).

Discretionary Access Control for devices in UTX/32S is somewhat different. There are three types of devices (in the DAC sense): privileged, allocatable (or owned), and discretionary. Privileged devices can only be accessed by privileged processes and only if the process has search access to the directory containing the device special file for that device.

Allocatable devices may only be accessed by privileged processes or processes executing on behalf of the owner of the device. These processes also need search access to a directory containing a device special file for that device, and read and/or write access (whichever is appropriate for the requested operation) to the device special file itself. Unprivileged processes gain access to allocatable devices through trusted servers. Trusted servers are privileged programs that have the ability to change the owner of the device to the user requesting access.

Discretionary devices can be accessed by any subject and are protected only by the protection bits associated with the device special file for that device.

Conclusion

UTX/32S Release 1.0 satisfies the C2 Discretionary Access Control requirement.

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 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

Storage Objects

Storage objects to which the Object Reuse requirement is relevant include: virtual memory, physical memory, and allocatable devices.

Virtual Memory

As explained in the system overview, UTX/32S is a demand-paged virtual memory system. A process address space comprises four areas of virtual memory: text; initialized static data; uninitialized data [static (bss) and dynamically allocated (brk)]; and the process stack. Text (i.e., read-only code and constants) and initialized data are stored on disk, while the remaining data and stack require main memory segments to be allocated for program execution.

When a process is created the kernel determines the uninitialized static data (bss) and stack space required (bss is included in the calculation of the data area for the file). Pages for these uninitialized data are not read from disk; rather available memory frames are found and mapped to the process, and the page table entries are marked as zero-fill. The bss and stack pages are then zero-filled when first referenced during process execution. If additional bss space is needed during execution, another memory page is allocated and zero-filled.

The kernel then allocates memory for initialized data and text pages and marks the page table entries fill-on-demand, indicating that they are to be filled from the disk file when needed. When text or data not in the swap area are referenced, the pagein routine fetches the page from disk. Swappable pages in the user address space include pages containing text, data, stack, and the user structure ("u"); pages in kernel space containing page tables also may be swapped. A pointer is contained in the process structure when the process is memory resident. When the process has been swapped out of main memory, the process structure contains the address of the process on the swap device. The process structure is kernel resident and can not be swapped out of main memory.

All data regions (initialized data, bss, and brk) are treated as a single data region within the kernel. To expand or contract the data region dynamically during execution, the process issues a break (sbrk) system call, with the amount of memory requested as the argument. The kernel expands the page table entries if necessary and marks them zero-fill-on-demand. Total allocated stack space is not allowed to grow during process execution.

During system initialization, the kernel ensures that no I/O is in progress and that no residual data previously read from I/O devices are thereafter treated as user data. The kernel also performs the following functions; it clears and frees user core and blocks the reception of incoming packets until protocols are initialized.

Recovery from a crash of the editor or system during an editing session is handled by exrecover, which restores the files being edited when the crash occurs. The files are saved by expreserve, which looks for temporary files both in the restricted and administrative environments (latter first) and moves them to a known location. As opposed to standard UNIX (which uses /usr/preserve), UTX/32S stores saved files in /usr/preserve/‹user› to ensure that malicious users cannot tamper with the data saved by expreserve. exrecover calls getpwuid to obtain the name of the directory to be searched for saved files. Once the last saved file is removed, the directory itself is also removed.

Physical Memory

During a context save, the CPU loads the new processor status double word (psd), blocks interrupts, and saves the base and general-purpose registers and stack pointers. During the restore, the CPU blocks interrupts, restores all general-purpose

and base registers from the stacks, and loads the saved psd. No residual data are left in registers in real memory for access by the new process. As for data in physical memory, as the new process is brought in, it overwrites any existing data within the accessible working set area, thereby making them inaccessible to the new process.

Only the system administrator may perform dump/restore operations and access archived information. Users must request special dump/restore services from the administrator.

Standard UNIX defines a pseudo-device /dev/mem which allows reading of physical memory. However, UTX/32S restricts the use of /dev/mem to privileged processes, and the write capability has been removed (i.e., it is read-only).

Allocatable Devices

Input and output to a device are handled as reading and writing a file dedicated to that device. All devices have device special files kept within the administrative environment.

Restricted devices (i.e., those which may be accessed only by the owner or a superuser) which may be allocated to a single user include login devices and user devices (e.g., magnetic tape drives, external communication lines). Trusted processes allocate and deallocate these devices and ensure that they are allocated to no more than one user at a time. Events relevant with regard to object reuse include allocation, deallocation, and writing to a device allocated to another user.

Login terminals and the system console are allocated for the duration of the login session. They are allocated when the user logs in and deallocated when he or she logs out. The write command was eliminated from the system since it was deemed inferior to the talk command.

The device control server is responsible for changing the ownership of all other allocatable user devices. Upon deallocation of a device, the device control server clears the device of user data before it is reallocated. In allocating a device, the server issues the setdevowner system call, which calls a kernel routine (forceclose in sys/sys/sys_inode.c), which forces all open file descriptors to be closed, thus preventing a user from being able to read or write to that device. If the device is a magnetic tape, the server will also issue an ioctl to rewind the tape device and take it off line.

Paragraph 3.24 of the System Administrator's Guide [3] provides recommendations for erasing magnetic tapes and disk packs before reuse.

Conclusion

UTX/32S Release 1.0 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

UTX/32S uses the separation between the administrative environment and the restricted environment to provide protected identification and authentication. There exists a set of identification, authentication, and capability (IAC) information for each defined user. The IAC information is controlled and protected by the TCB within the administrative environment in the files /etc/passwd and /etc/group. Each user is correctly identified and authenticated by use of the login process, which is invoked only by the getty process started by the init process. All three processes are run in the administrative environment and check that they were called by a process with superuser privileges. The login process checks that the authenticating password, when encrypted, matches the encrypted password stored with the IAC data, and then associates the logging ID with the allowed capability. The capability information consists of either the five-character field "admin" or a blank field depending on whether or not the user is allowed to log into the administrative environment. By means of a user ID associated

with each process, which is set to the logging ID of the user who invoked it, each authenticated user is continuously associated with all processes acting on the user's behalf.

The IAC information for each user contains: name and logging IDs, password (encrypted), capability information, group memberships, home directory and login shell. The group information is stored in the /etc/group file and the other six items are stored in the /etc/passwd file.

Access to the IAC information is restricted to the superuser, and administrative personnel must use utilities /etc/vipw and /etc/vigr to modify the corresponding files. The default editor vi may be replaced by any other editor that operates in the administrative environment. The restriction to the use of vipw and vigr ensures the integrity of these files, while the ability to choose which editor to use allows for ease of maintenance. A user may change the password stored in the /etc/passwd file by invoking the passwd process, which uses the authorization server to cause the encrypted password to be changed. This server is not interruptible, and invoking it and then attempting to interrupt it will not allow access to the administrative environment.

The IAC information in the administrative environment is not propagated to any other system component. Some unprivileged user utilities require access to the IAC information. The utilities /etc/updatepw and /etc/updategr automatically produce sanitized copies of the password and group files within the restricted environment, storing them in /renv/renv1/etc/passwd and /renv/renv1/etc/group. Information that can be observed from within the restricted environment includes user ID, logging ID, group membership, and restricted environment (home directory and login shell).

Each process has a user ID associated with it which identifies the privileges with which it was invoked. The user ID is the same as the logging ID for the initial login process. The logging ID of a child process of a trusted server process is set to the logging ID of the user who requested the service. Processes are limited to the capabilities determined for the invoking user at the time of authentication. The base login process for each user is created immediately following authentication. The newly created process is confined to the restricted environment specified by the user during login. Each user's activities are attributable to the corresponding logging ID, which makes the user accountable for the actions of all invoked processes. Whereas other versions of UNIX allow a user to call login during a session and possibly change the

environment in which his processes run, UTX/32S does not allow login to be used as a program. In fact, only a system process may call login. The intention is that login be called by getty which is called by init as soon as a terminal connection is physically opened.

The procedures connected with the login process allow a system administrator to configure the system such that most of the recommended practices in the Password Management Guideline [2] may be followed. In particular, UTX/32S enforces a 6-character minimum password length, at least one character must be non-numeric, and only the first 8 characters of a password are recognized.

Conclusion

UTX/32S Release 1.0 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

Auditing in UTX/32S is initiated by a system command, setslogfile [filename], which names the file in the administrative environment where the audit trail data will be written. A system administrator may turn off all audit logging by the command setslogfile -s. The setslogfile command is normally invoked when the system is brought up in single user mode if any audit logging is to be performed. This command must be issued because the mandatory events, which are always logged when auditing is enabled, are required to properly analyze the audit trail. Other mandatory logging records are produced by events which require superuser privilege. Optional event logging may be specified by calling setslogdetail with a list of optional events or classes of events as its argument.

There are 27 event types that are generated by calls from kernel routines to the kernel routine securelog, which writes a message to the audit trail file, provided the routine was called by a process that was invoked by the superuser. There are also 6 event types associated with trusted servers that are generated by a call by a trusted server to the system call entry to securelog in the administrative environment, which then calls the kernel routine securelog.

The mandatory kernel events are:

- changing accounting information about a user.
- changing a user's current working directory.
- changing the name of a device owner in the table within the kernel.
- setting a user's logging ID, controlling tty, and restricted environment ID during login.
- setting the log ID of a process.
- changing the audit trail logging mask (specification of optional logging events).
- creation of a special file using the mknod kernel routine.

- mounting a file system.
- unmounting a file system.
- rebooting or shutting down the system.
- starting or stopping the audit trail.
- process exit.
- process fork.
- requests for superuser status from administrators.

Optional events are:

- unsuccessful attempts to write a file owned by the root.
- unsuccessful attempts to access a device.
- unsuccessful attempts to access a root-owned device.
- unsuccessful attempts to access a file due to lack of correct discretionary access permission.
- creation of hard or symbolic links.
- removal of hard or symbolic links.
- successful attempts to access the current audit trail file.
- creating or removing a directory.
- opening a file (note that this corresponds to a successful access of a file).
- renaming a file.
- change of the discretionary access modes of a file or directory.
- changing the ownership or group of a file or directory.
- it is an optional event to log the system version, CPU type, configuration of the SelBUS(1) and IOP devices, and current memory configuration when audit logging is enabled.

The mandatory events associated with trusted processes are:

- requests by any user or administrators for superuser privileges.

- all uses of login, whether successful or unsuccessful. The login process was removed as a built-in command, and must be invoked by a process, getty, with superuser privilege.

The optional trusted server events are:

- attempts to suppress the burst page on the lpr command by using the -h argument.

- use of any command that attempts to access the /etc/passwd or /etc/group files in the Administrative Environment.

- unsuccessful login attempts, which also involve a trusted server, are logged, including the logging ID given and the controlling tty.

- any attempt to change the name identifying the sender of mail.

Each audit record includes a time stamp, the process ID that caused the record to be generated, the user ID and logging ID of the user that invoked the process, and the restricted environment in which the process was executing. The identifier of the terminal associated with the process and a message associated with the event being logged are also included. For file access events, the access modes of the file, the owner of the file, the group ID and name of the file, the inode number of the file, the current working directory when the access was attempted, and the arguments typed by the user attempting the access are recorded.

In addition to protecting the audit trail file within the administrative environment, sufficient controls are placed on this file to ensure that no audit records are lost due to insufficient space on the audit device. When the audit trail file reaches 95% of its capacity, warning messages are sent to the console device. If these warning messages are ignored and the audit file becomes full such that no additional audit records can be written, processes will be put to sleep whenever an action that generates an audit record is attempted. Also, a five minute countdown is started that ends in a system shutdown unless a system administrator either turns off auditing or provides more space for the audit trail file.

Reports may be generated from the collected audit data using the sanalyze utility. This allows anyone with appropriate access to the audit files (i.e., administrators outside of the restricted environment) to extract information based on specific selection criteria. These criteria including logging ID, event types, and time periods. Both mandatory and optional events may be specified by the sanalyze command. Since setslogdetail can only specify logging of events by all users, sanalyze must be used to filter the audit trail file in order to selectively audit the actions of any one or more users based on individual identity. Note: sanalyze will not correctly report on files which are accessed via symbolic link (i.e., the pathname of the link will be included in the audit file instead of the pathname to the target file). However, sufficient audit messages are generated to allow a reconstruction of any disallowed access attempts. This limitation is inconvenient but still acceptable.

While setslogfile may be called at any time to initiate recording of the minimum audit data necessary for sanalyze to analyze the audit trail, a complete and correctly analyzable audit trail will be recorded only if setslogfile is called in single-user mode before switching to multi-user mode. This is the default condition as a result of /etc/rc, which executes at system startup, containing an appropriate call. This is properly noted in the System Administrator's Guide [3].

Due to encoding of users and groups by assigning them integer codes in place of their original characters, sanalyze must have access to these codes as stored in the /etc/passwd and /etc/group files. If sanalyze is run using different versions of these files from the versions in use when the audit trail file was generated, the output of the audit log reduction will be incorrect. This is properly documented in the System Administrator's Reference Manual [4].

Conclusion

UTX/32S Release 1.0 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

The filestore of the UTX/32S system is divided into the Administrative Environment (AE) and the Restricted Environment (RE). The files which compose the TCB reside in the AE. These files include kernel files, privileged (root-owned) executables and data files, trusted servers and device special files. General (unprivileged) users are confined to the RE, making it the primary domain separation mechanism in the system. The RE is a sub-tree in the UTX/32S file hierarchy created by the chroot system call. The RE is structured so that the root directory ("/") for a process in the RE is not the root directory for the system (e.g., the evaluated system uses /renv/renv1 as the root for its restricted environment). Security relevant files can be accessed from within the RE only by means of a designated secure socket which communicates with an executable (called a trusted server) outside the RE. Confined processes are also not allowed to cd (change directory) out of the RE. A privileged process, confined to a restricted environment by the use of the setuser system call, loses its privileges. For example, when first logging on, the login program is privileged (i.e., has environment ID=0), but when it forks and execs the shell, the setuser call assures the user ID and the environment ID are non-zero. Therefore, a superuser process cannot exist in the RE. This is not to say that an administrator, while in the RE, cannot become (or be acting as) the superuser; this merely says that no process which is created in the RE can become the superuser. Therefore, subjects operating within an RE cannot modify the TCB or its data structures, and cannot access them except through a secure mechanism called a secure socket.

Subjects in UTX/32S are the processes which act on behalf of the users of the system. These processes are structured so that the lower 2MB of all processes' address space are identical; this lower 2MB is mapped directly to the kernel. A process can execute in the privileged or unprivileged state, as determined by the sixth bit of the PSW (see page 8, "System Overview, Subjects" for additional information). When the privileged bit is set, a system call is being performed for the process. Aside from the lower 2MB of the process, all other addressing structures and data areas are separate. The one possible exception is the text section of the process, text being the UNIX term for executable

instructions. This section consists of either actual instructions, or in the case that the text is shared among many processes, a pointer to the executable text, which is located somewhere in main memory. In either case, the text is read-only, and the data areas for all processes are separate. A unique logging ID is also maintained for each process for auditing purposes, and two other IDs (User ID and Environment ID) are used for access control purposes.

Process isolation is accomplished by each process having a per process data area known as the "u area" or "u block." This structure contains information such as the root directory for the process, the current working directory, the user and group IDs, plus a variety of pointers, to things such as the process' process table entry and the open inode (file) table. Each process also has a separate stack for the process data and frames for function calls. As mentioned above, all data areas for the processes are distinct. The physical address spaces for the data and user stack are also distinct, although the text may be shared. The u area is kept in the kernel address space, and copied on context switch.

As discussed previously, the filestore on UTX/32S is organized into two sections: the RE and the AE. Each file in the filestore is associated with an inode, which is a data structure that contains information about the file, such as owner, access modes, and where it is located on the disk. There are two types of links which may be made to these files: hard links and symbolic links. A hard link is indistinguishable from the original directory entry; any changes to a file are effective independent of the name used to reference the file. Hard links may not span file systems and may not refer to directories. In UTX/32S, hard links should not be made to files outside the RE. A symbolic link contains the name of the file to which it is linked. The referenced file is used when an open operation is performed on the link, and a stat on a symbolic link will return the linked-to file. Symbolic links may span file systems and may refer to directories. Devices and directories also look like files to the subjects. This being the case, access control and auditing become a matter of controlling the access to and auditing the use of files. All files are isolated by means of the inode list in the superblock in the UTX/32S kernel, every file having a distinct inode.

UTX/32S uses the Restricted Environment mechanism to maintain a separate domain for the TCB. Processes acting within the RE cannot tamper with the TCB, and cannot access any code or data belonging to the TCB except through the use of the trusted server and secure socket mechanisms. Resources controlled by the TCB

are treated as files; files are isolated by being associated with a unique inode. Processes are isolated by having separate user data and stack spaces.

Conclusion

UTX/32S Release 1.0 satisfies the C2 System Architecture requirement.

System Integrity

Requirement

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

Applicable Features

A set of tests is available to verify the correct functioning of the PowerNode hardware and firmware. This set of tests is known as the Gould Virtual Memory Diagnostic Facility. This set of tests is broken down into two collections of diagnostic routines, nucleus diagnostics, also known as LEVEL ONE diagnostics, and peripheral diagnostics, also known as LEVEL TWO diagnostics. The nucleus diagnostics include tests for the correct operation of the CPU/IPU functions, the SelBUS, and the internal clocks. The peripheral diagnostics test for the correct operation of the peripheral devices and their associated controllers. These diagnostic routines can be run either automatically or interactively.

VMCPUTST is a command file that automatically executes LEVEL ONE and LEVEL TWO diagnostics when the Diagnostic Facility is loaded and initialized. VMCPUTST executes a subset of all the available diagnostic routines. Specifically, it executes the following diagnostics:

LEVEL ONE - Control Store Loader
Central Processing Unit (Parts 1, 2 and 3)
Base Register
Interrupt
Trap
IPU Trap

Cache/Shadow Memory
Mapped Memory Management
Control Store
Effective Address
IOP Console

LEVEL TWO - Multiport Memory
Interval Timer

VMCPUTST halts or suspends execution of the diagnostic if an error is encountered. Operator intervention is then required to either proceed with the diagnostics, restart from the last diagnostic, or terminate the diagnostic test.

Each diagnostic can also be run individually by setting General Purpose Register Zero to some non-zero value before loading and initializing the diagnostic facility. LEVEL ONE diagnostic routines are stand-alone programs while LEVEL TWO diagnostics are programs which must be loaded and executed under a Diagnostic Executive Program (DEXP).

All errors encountered by the diagnostic routines, whether they are run automatically or individually, are recorded on the Listed Output (LO) file.

Conclusion

UTX/32S Release 1.0 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 conducted testing during the fall of 1986. The test system was a Gould PowerNode 6000 running UTX/32S, Release 1.0.

The first step in the testing process was for the evaluation team to generate a system from binary (i.e., object code) tapes. The team followed the procedures outlined in the System Installation Guide. Upon successful generation and booting of UTX/32S the team installed and executed Gould's Security Test Suite (STS). During the course of testing, the NCSC formal evaluation team executed all of automated and semi-automated tests in the STS. Due to time constraints, the evaluation team executed only 92% of the manual tests contained in the STS.

The evaluation team developed shell tests to exercise some areas of the system more extensively. Shell scripts were written to test the umask mechanism, object reuse on all file system elements, and file system element creation. Attempts were made by the evaluation team to gain unauthorized access to the audit logs, and the file /etc/passwd through the use of system calls. Code for /etc/rc was examined to ensure that an audit log was properly created during system initialization.

The development of Gould's Security Test Suite allowed Gould to locate six implementation flaws in UTX/32S. These flaws were corrected.

Conclusion

UTX/32S Release 1.0 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

Volume 1 of the Gould UTX/32S Document Set, User's Guide and User's Reference Manual, contains a document entitled Using Gould UTX/32S [7] which fulfills this requirement. This document is addressed to readers assumed to be knowledgeable users of UNIX-based operating systems. Protection mechanisms and guidelines for their use are described relative to the UNIX-based Gould UTX/32 system. References to tutorials and other texts are provided for readers unfamiliar with UNIX-based systems.

This document provides a good overview of the security mechanisms provided by UTX/32S, and a clear and concise discussion of sound security practices for users. Procedures and practices for secure login and logout are described, including warning signs of possible security violations which should be reported to the system administrator. Password selection and protection guidelines are provided, as well as guidelines for protecting files.

The protection mechanisms provided by the TCB, including file protection mechanisms, the restricted and administrative environments, trusted servers, and device controls, are discussed in terms of commands and routines which have been deleted or whose functionality has been changed relative to other UNIX-based systems. The role of the umask mechanism in file protection is described. Although the audit trail is actually used only by the system administrator, its existence is mentioned so that users are aware that their actions are subject to auditing.

The document explains to users that they are automatically placed in a restricted environment, while access to the administrative environment is limited to users designated as administrators. It also explains that mail, news, and msgs are kept within the administrative environment, with user access via trusted servers. Access policy and procedures with regard to allocatable devices (terminals, tape drives) are discussed, including mention of the fact that access controls are enforced by the kernel.

Conclusion

UTX/32S Release 1.0 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 criterion for a Trusted Facility Manual (TFM) is met by the Installation Guide [5] and the System Administrator's Guide [3], both contained in Volume 3 of the Gould UTX/32S Document Set. The System Administrator's Reference Manual [4], and the Software Release Notes [6], also contained in Volume 3, supplement these guides.

Both of these guides are specifically addressed to the system administrator. The purposes, goals, and scope are stated for each, and pointers to related information are provided. Chapter 2 of the System Administrator's Guide, entitled "Security and the UTX/32S System," provides a good overview of security as implemented in UTX/32S and discusses the TCB mechanisms and how protection is enforced. Both the Installation Guide and the System Administrator's Guide include SECURITY NOTES, which emphasize information essential to installing and operating a system so that the security features are maintained. The Installation Guide stresses that the installation steps must be followed precisely in order to attain the C2 evaluated configuration and warns that any deviations will nullify the rating. It also advises the system administrator to read both "Security and the UTX/32S System" and Appendix C, "Security Checklist," of the System Administrator's Guide before beginning installation. The "Security Checklist" provides guidelines to be followed in order to maintain the system as a C2-rated system.

The TCB is defined to include the operating system, system hardware, and privileged processes -- all of which must be trusted. For each of these elements of trust a section discussing responsibilities is included. Elements of the operating system included in the TCB are: the kernel; the administrative environment; and superuser, system, and administrator processes.

A good overview of the security intended to be enforced by the system and of the system administrator's role in protecting objects is provided. The System Administrator's Guide stresses good security practices such as password management and monitoring of user activities. It also discusses potential security vulnerabilities, such as dial-in lines and Trojan Horse attacks targeted at administrator programs running in the Restricted Environment, and describes precautions to be taken to minimize such vulnerabilities. Throughout both the Installation Guide and the System Administrator's Guide, SECURITY NOTES highlight and emphasize actions which could nullify the system's C2 rating.

Differences between operating in single-user mode versus multi-user mode are described, as is the process of adding users to the system. In discussing how to add users, the System Administrator's Guide tells the system administrator to set the default umask to 077 in the user's home directory to ensure that files created by the new user are not accessible to other users unless the new user explicitly allows such access.

Physical protection is addressed where applicable, such as for removable media (i.e., disk packs and tapes). Methods of erasing data from removable media before reuse also are described. Appendix D of the System Administrator's Guide provides a good summary of the security features of UTX/32S which can be used by the system administrator to make the entire system more secure, along with pointers to applicable sections of documentation for details.

The System Administrator's Guide discusses the audit capabilities of UTX/32S. It references the System Administrator's Reference Manual for a complete and very detailed description of sanalyze, the audit reduction tool. The reference manual pages enumerate events which are audited by default and those which are optional, and provides a detailed audit record structure and an example which should be useful to the SA in performing system audit. Other relevant and useful reference manual pages are referenced, specifically: setslogdetail, setslogmask, and setslogfile.

Conclusion

UTX/32S Release 1.0 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

Gould's Security Test Suite (STS) is documented in the following three documents.

- Guide to the Security Test Suite for Gould UTX/32S, Release 1.0 [12]
- Security Policy Model for Gould UTX/32S, Release 1.0 [13]
- UTX/32S 1.0 Security Test Suite Source Listing [14]

These documents contain the information necessary to run, maintain, and update the STS.

The Guide to the Security Test Suite for Gould UTX/32S, Release 1.0 provides an overview of the Security Test Suite and instructions for executing the tests.

Tests for UTX/32S are designed to show that the system implements a particular security policy statement. The Security Policy Model for Gould UTX/32S, Release 1.0 identifies the security policies that UTX/32S was designed to enforce. This model identifies policies in a hierarchical manner. Generic policies are restated in successively more implementation-specific policies. Finally the implementation specific policies are exercised by tests.

Documentation in the source listings identifies the security policies that are examined by the test, the testing procedures, and the goal of the test. Many but not all of the tests identify the expected result for each test case. Those tests which do not explicitly identify the expected results provide sufficient information to allow the reader to infer the expected results.

Conclusion

UTX/32S Release 1.0 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 following documents were provided to the evaluation team in fulfillment of the Design Documentation criterion:

- Security Policy Model,
- Program Maintenance Manual UTX/32S, Release 1.0 (DRAFT),
- "System Calls" and "Maintenance" in the System Administrator's Reference Manual,
- 4.2BSD and UTX-32 Differences Study for Gould UTX/32S, [16]
- UTX/32S Traps and Interrupts, [17]
- Memory Management for Gould UTX/32S, [18]
- Object Reuse Study for Gould UTX/32S. [19]

The Gould UTX/32S 1.0 Security Policy Model describes Gould's philosophy of protection and explains how this philosophy is translated into the TCB. It identifies all elements comprising the TCB, including the kernel, programs, data files, and processes. Subjects and objects are identified, and the mediation of accesses between them is described. A mapping from the TCB to the security philosophy is provided, and the discretionary access control, identification and authentication, and audit features and mechanisms are described. Additionally,

the document discusses the role of secure sockets in interprocess communications. The Gould UTX/32S 1.0 Security Policy Model identifies all programs comprising the TCB.

The kernel interface is described by the "System Calls" section of the System Administrator's Reference Manual. The "Maintenance" section of the reference manual comprises man pages useful for systems programmers in maintaining UTX/32S.

4.2BSD and UTX-32 Differences Study for Gould UTX/32S describes differences between 4.2BSD UNIX and Gould UTX/32 1.2. Using "4.2BSD and 4.3BSD as Examples of the UNIX System," by J. S. Quarterman, A. Silberschatz, and J. L. Peterson [Computing Surveys, Vol. 17, No. 4, December 1985, pp. 379-418], as a baseline, the document identifies all instances where Gould UTX/32 differs from the described UNIX system.

The UTX/32S Program Maintenance Manual describes code modifications made to UTX/32 to meet the requirements of the Gould UTX/32S 1.0 Security Policy Model. The document includes an overview of the mechanisms implemented in UTX/32S to strengthen security and to correct problems found in UTX/32 and other UNIX systems, and detailed descriptions for: the implementation of trusted servers to replace the functionality of the eliminated setuid and setgid bits; kernel modifications; auditing mechanisms; and additions, deletions, and modifications to utilities and libraries. Each module description includes an overview, a functional specification, and a design specification. Pointers to source code, which Gould made available to the evaluation team, are provided.

Security-critical features of the Gould PowerNode hardware used by UTX/32S are described in UTX/32S Traps and Interrupts and Memory Management for Gould UTX/32S. UTX/32S Traps and Interrupts describes how UTX/32S makes use of the trap and interrupt facilities to interface with the hardware and process environments. Memory Management for Gould UTX/32S describes how UTX/32S uses the memory management facilities of the PowerNode hardware to provide the process environment. Both documents include applicable material from Gould SS 6 (Virtual Mode), V6, and V9 Central Processing Unit Reference Manual.

Object Reuse Study for Gould UTX/32S provides details regarding how UTX/32S hardware and software manage system objects. This study identifies the system resources which can be allocated and deallocated, and details the strategies used to ensure that one process cannot gain access to the resources or data previously allocated to another process. This study, along with Memory

Management for Gould UTX/32S, provides a good description of UTX/32S design features which are used to meet the Object Reuse criterion.

Conclusion

UTX/32S Release 1.0 satisfies the C2 Design Documentation requirement.

This page intentionally left blank.

EVALUATORS' COMMENTS

This section consists of comments and/or opinions from the evaluation team in the following areas:

- features provided by the system (or vendor) that are not required by the Criteria.
- features provided by the system (or vendor) that are required by the Criteria but at a level higher than this evaluation was focused.
- the useability of some of the features that satisfy the Criteria requirements.

Configuration Management

Gould's configuration management system controls changes to system software, the security test suite, and documentation. Configuration management of system software includes modifications to UTX/32 and new software developed for UTX/32S. New software for UTX/32S is the software that implements the restricted environment, utilities in the restricted environment and secure servers. The software changed from UTX/32 are the kernel modifications to support secure sockets and deletion of setuid, and utilities that formerly used setuid.

The documents under configuration control are listed below.

- Gould UTX/32S User's Guide, Release 1.0
- Gould UTX/32S User's Reference Manual, Release 1.0, Volumes 1 and 2
- Gould UTX/32S Installation Guide, Release 1.0
- Gould UTX/32S System Administrator's Guide, Release 1.0
- Gould UTX/32S System Administrator's Reference Manual, Release 1.0
- Gould UTX/32 Assembler Reference Manual
- Gould UTX/32 Input/Output Subsystem Manual

- Gould UTX/32 Fortran 77 Reference Manual
- Introduction to Gould UTX/32S Documents
- Gould UTX/32S Software Release Notes, Release 1.0

The control of source code for the UTX/32S operating system is handled by an automated tool, the Revision Control System (RCS). RCS is capable of identifying the line-by-line changes between any two versions of a software module or document. RCS is also capable of generating the baselined versions of the system.

Gould has a bug-tracking system in place that tracks the correction of reported bugs. When bugs are reported they are assigned to developers to be corrected. The fix is reviewed and audited for its effect on security. The security audit is performed by the senior technical staff (i.e., the project scientist) of the project. This audit involves code review and/or testing of the program to determine whether the problematic behavior of the program has been corrected by the proposed fix. The audit also ensures that the fix has no detrimental side effects on the security of the system. At the completion of an audit the project scientist checks the revised code into RCS, for inclusion in the next baseline.

When new baselines are created RCS identifies those modules that have been changed since the last baseline. Modules that have changed must identify a software problem report (SPR) or design document that describes the changes and other modules affected.

Although Configuration Management is not a requirement of the Criteria until class B2, Gould has in place a configuration management system that encompasses their source code, documentation, and test suite. If evaluated today against the B2 requirements, Gould's current configuration management plan would not be found sufficient. However, it could easily be enhanced to be so.

Informal Security Policy Model

Although there are no requirements for any type of security policy model at the C2 class, Gould has prepared an informal security policy model for UTX/32S. This informal model describes the types of subjects and objects present in the UTX/32S system and the rules governing how these subjects can access objects. However, the evaluation team did not make any judgements as to

whether the informal model for UTX/32S was sufficient to satisy the B1 requirements for an informal model. Nonetheless, the team feels that it is a useful document.

Single User Granularity in DAC

To provide access control to the granularity of a single user in UTX/32S, an administrator must define the group structure such that all possible combinations of groups exist. This is not a very convenient way to use the discretionary access control mechanism and it is not envisioned that this will be done at many, if any, sites. This problem, however, is common to all protection bit based access control mechanisms and is not a problem specific to UTX/32S.

Default Discretionary Access

UTX/32S does possess the capability to provide default access control on all newly created objects. This capability, however, is not correctly configured in the system as delivered by Gould. A discussion of how an administrator can properly configure the system to provide this capability is provided in the Trusted Facility Manual (see page 32, "Evaluation as a C2 System, Trusted Facility Manual").

This page intentionally left blank.

EVALUATED HARDWARE COMPONENTS

The hardware covered by this evaluation includes all processors in the PowerNode product line. The primary requirement for hardware evaluation is that the hardware function properly. This is verified by the system integrity tests (see page 28, "Evaluation as a C2 System, System Integrity") and was not given a detailed reevaluation by the team. The integrity assurances provided by the Gould-supplied diagnostic tests are satisfactory.

List of Evaluated Components

This section lists, by category, the Gould 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

6006	PowerNode 6000 Basic System
6032	PowerNode 6030 Desk High System
6040	PowerNode 6040 Processor (10 main bus slots)
6041	PowerNode 6040 Processor (18 main bus slots)
6051	PowerNode 6050 Basic System
6081	PowerNode 6080 Basic System

NOTE: For the following CPU Boards in the PowerNode 6000, the indicated release level (or later) is required:

Cache SelBus (CS)	160-103-655-001C
Instruction/Execution (IE)	160-103-656-001C
Microstore (MS)	160-103-654-001D

9003	PowerNode 9000 Basic System
9052	PowerNode 9050 Basic System
9082	PowerNode 9080 Basic System

NOTE: For the following CPU Boards in the PowerNode 9000, the indicated release level (or later) is required:

L3	160-103-846-001C

Processor Options

For the PN6000 Series:

3611	Hardware Floating Point Accelerator
7906	Virtual IPU Conversion Kit
7907	Virtual IPU Conversion Kit

For the PN9000 Series:

3826	32KB Cache Memory Upgrade
3828	Multiply Accelerator
3829	Multiply Accelerator Field Upgrade

Memory Options

For the PN6000 Series:

3034-2	2MB ISM
3034-4	4MB ISM
3035A	8MB ISM

For the PN9000 Series:

3028	1MB IMM
3034-2	2MB ISM
3034-4	4MB ISM
3035A	8MB ISM Expansion Package

Communication Products

8512-2	Eight-Line Asynchronous Controller
8580	Distribution Panel RS-232C
8581	Distribution Panel RS-423
8582	Distribution Panel Current Loop
8585	RS-232C Distribution Panel, 9 in.
8586	32-Port RS-232C Distribution Panel

I/O Interfaces

8001	I/O Processor
8031	Line Printer/Floppy Disk Controller
3050	High-Speed Tape Processor

8051	Buffered Tape Processor
8060	Universal Disk Processor
9020	Low-Speed Tape Processor

Disk Products

8138	80MB Mini-Cartridge Disk
8139	80MB Mini-Cartridge Accesories
8191	Disk Processor Eight Drive Option
8830	80MB Removable Disk Processor Subsystem
8836	80MB Mini-Cartridge Disk Processor Subsystem
8840	300MB Removable Disk Processor Subsystem
8856	340MB Mini-Winchester Disk Processor Subsystem
8858	Additional 340MB Mini-Winchester Disk
8880	Dual 5.25-inch Floppy Disk Package
9342	80MB Removable Expansion Disk Drive
9346	300MB Removable Expansion Disk Drive
9349-3	80MB Disk Pack
9349-4	300MB Disk Pack

Magnetic Tape Products

8214	125 IPS Tri-Density HSTP Subsystem
8224	125 IPS Slave Tri-Density Mag Tape Unit
8250	Streaming Tape Subsystem
8251	6250 BPI Streaming Tape Subsystem
8255	Streaming Mag Tape Unit
8256	6250 BPI Streaming Tape Unit
9364	45 IPS Slave Mag Tape Unit
9378	75 IPS Slave Mag Tape Unit
9576	75 IPS 800/1600 BPISLTP Subsystem
9578	45 IPS 800/1600 BPISLTP Subsystem

Cabinets/Chassis/Power Supplies

For the PN6000 Series:

2374	Battery Back-up Unit
8192	Mini-Disk Mounting Kit
8910	I/O Expansion Chassis

For the PN9000 Series:

8192 Mini-Disk Mounting Kit
8910 I/O Expansion Chassis

Test System Configuration

The system that the functional tests were performed on was configured as follows:

- PN6000 series processor
- Floating Point Accelerator (Model 3611)
- 16MB ISM memory (Model 3034)
- Universal Disk Processor (Model 8060)
- One 80MB disk (Model 8830) and one 300MB disk (Model 8840)
- Low Speed Tape Processor (Model 9020)
- I/O Processor (Model 8001)
- Line printer controller (Model 8031)
- Asynchronous Communications Controller (Model 8512)
- Data Products line printer (Model 9225)
- Pertec tape drive (Model 9020)

EVALUATED SOFTWARE COMPONENTS

UTX/32S Trusted Computing Base Software

The Security Policy Model for Gould UTX/32S, Release 1.0 identifies the Trusted Computing Base Software of UTX/32S. The TCB is categorized in four parts: 1) the operating system kernel, including all I/O device drivers, 2) all programs that reside outside all restricted environments, 3) all data files whose contents determine the outcome of a security decision by any TCB subject, 4) all privileged processes.

The following subsections identify TCB data files and define the utilities and system files found in a UTX/32S system. Those utilities inside the restricted environment form a subset of those in the administrative environment. Administrators require all utilities that unprivileged users require, but administrators also have access to purely administrative commands and to commands that require superuser privileges to execute.

TCB Data Files

All data files whose contents determine the outcome of a security decision by any TCB subject are a part of the TCB. There may be other files used by each trusted program. Each is documented in the UTX/32S 1.0 reference manual.

/etc/passwd	IAC information
/etc/group	IAC information
/etc/rc	system startup activities
/etc/rc.local	system startup activities
/etc/fstab	file system description
/usr/lib/crontab	data file for cron (8)
/etc/tiptab	data file for tip (1)

TCB Programs

Parentless Process

/etc/init	system initialization

Trusted Servers

/etc/auth_s	Authorization Server
/etc/admin_s	Administrative Server
/etc/devctrl_s	Device Control Server
/etc/generic_s	Generic Server
/etc/lpr_s	Line Printer Server
/etc/mail_s	Mail Server

Other

/etc/updatepw
/etc/updategr
/bin/sh

Contents of Administrative Environment Only

The administrative environment contains executables, directories and data files. For all lists in the remainder of this Appendix, except for /dev, the following conventions will be used:

* indicates an executable file

/ indicates a directory

All other names are those of data files.

For /dev (in both the Administrative Environment Only list and the Administrative and Resticted Environment list) the following conventions are used:

? a "wildcard" character used to represent any single character

a "wildcard" character used to represent any number of characters, including none.

[] indicates that each character contained within the brackets is to be used to complete the full name. For example, xx[a-c] represents xxa, xxb and xxc.

NN represents numerical digits ranging from 01 to 99

/bin

acctcom*	ps_utx*	sush*
df_utx*	sucsh*	wall*
login*		

/etc

acs/	grpck*	rc.local*
admin_s*	halt*	reboot*
analyze*	hosts	repquota*
arff*	icheck*	restore*
auth_s*	init*	sanalyze*
catman*	kgmon*	savecore*
checkall*	link*	services
chown*	lpc*	setslogdetail*
chroot*	lpr_s*	setslogfile*
clri*	mail_s*	shutdown*
comsat*	mkfs*	swapon*
cron*	mklost+found*	syslog*
dcheck*	mknod*	syslog.conf
devctrl_s*	mkproto*	syslog.pid
diskpart*	motd	system_setup/
dmesg*	mtab	talkd*
dump*	ncheck*	tiptab
dumpdates	networks	ttys
dumpfs*	newfs*	ttytype
edquota*	prep*	tunefs*
fastboot*	printcap	umount*
fasthalt*	protocols	unlink*
flcopy*	pstat*	update*
flformat*	pwck*	updategr*
fsck*	quot*	updatepw*
fstab*	quotacheck*	utmp
generic_s*	quotaoff*	vigr*
getty*	quotaon*	vipw*
gettytab*	rc*	wtmp

/lib

None.

/usr/bin

install*	mesg_utx*	non-btl*
iostat*		

/usr/lib

acct/	lpd*	renvname*
adb/	lpf*	sendmail.cf
crontab*	lrf*	sendmail.hf
getNAME*	makewhatis*	sendmail.st
libsys.a	necf*	whatis
logger*		

/usr/ucb

biff_utx*	lprm_utx*	vmstat*
gcore*	mailq	w*
last*	newaliases	w_utx*
lastcomm*	quota*	whodo*
lpq_utx*		

/dev

dk	ioi[0-3]	mem
drum	kUmem	printer
fl*	kmem	

Contents of Administrative and Restricted Environment

/bin

[*	ed*	pr*
adb*	expr*	ps*
ar*	false*	pwd*
as*	grep*	rm*
awk*	hostid*	rmdir*
cat*	hostname*	sed*
cc*	kill*	sh*
chgrp*	ld*	size*
chmod*	ln*	strip*
cmp*	ls*	stty*
cp*	mail*	sync*
cpio*	make*	tar*
csh*	mkdir*	tar_1024*
date*	mt*	tee*
dd*	mv*	test*
df*	nice*	time*

diff*	nm*	tp*
du*	od*	true*
e*	pagesize*	who*
echo*	passwd*	

/etc

bcopy*	mount*	remote
config*	mvdir*	renice*
group	passwd	termcap
modcap	phones	

/lib

c2	cpp	libc.a
ccom	crt0.o	mcrt0.o
codgen	f77pass1	

/usr/bin

addbib*	find*	regcmp*
admin*	fspan*	rev*
asa*	fsplit*	rmdel*
basename*	get*	roffbib*
bc*	getopt*	sact*
bdiff*	help*	sccsdiff*
bfs*	hyphen*	sdiff*
bs*	id*	sleep*
cal*	indxbib*	sno*
calendar*	join*	sort*
cb*	learn*	sortbib*
cdc*	lex*	span*
checkeq*	line*	spell*
checkmm*	lint*	spellin*
col*	logname*	spellout*
comb*	look*	spline*
comm*	lookbib*	split*
csplit*	lorder*	struct*
cu*	m4*	style*
cut*	mesg*	sum*
cxref*	mm*	tabs*
dc*	neqn*	tbl*
delta*	newform*	tip*
deroff*	news*	touch*
devctrl*	nl*	tr*
diction*	nohup*	tsort*

diff3*
diffmk*
dircmp*
dirname*
efl*
egrep*
env*
explain*
f77*
fgrep*
file*
nroff*
osdd*
pack*
paste*
pcat*
prof*
prs*
ptx*
ranlib*
ratfor*
refer*
tty*
uname*
unget*
uniq*
units*
unpack*
val*
vc*
what*
xargs*
yacc*

/usr/lib

Mail.help
Mail.help.~
Mail.rc
aliases*
aliases.dir*
aliases.pag*
calendar*
dict.d*
diff3*
diffh*
dprog*
eign*
ex3.7preserve*
ex3.7recover*
explain.d*
gcrt0.o*
gprof.callg.blurb*
gprof.flat.blurb*
help/
learn/
lex/
lib.b
libF77.a*
libI66.a*
libI77.a*
libPW.s*
libU77.a*
libc_p.a*
libcurses.a
libcurses_p.a
libdbm.a*
libg.a*
libioi.a
libl.a*
libln.a*
libm.a*
libm_p.a*
libmp.a*
libnm.a*
libnm_p.a*
libtermcap.a*
libtermcap_p.a*
libtermlib.a*
libtermlib_p.a*
liby.a
lint/
macros/
me/
more.help*
ms/
refer/
sendmail*
spell*
struct/
style1*
style2*
style3*
suftab
tabset/
term/
tmac/
units*
vacation*
xcpp*
xpass*
yaccpar*

/usr/ucb

Mail*
apply*
aproprs*
arcv*
biff*
ccat*
checknr*
gprof*
grep*
groups*
head*
indent*
leave*
lock*
strings*
symorder*
tail*
talk*
trman*
tset*
ul*

chsh*	lqp*	uncompact*
clear*	lpr*	unexpand*
colcrt*	lprm*	uptime*
colrm*	mail*	users*
compact*	man*	vi*
ctags*	mkstr*	view*
dbx*	more*	wc*
e*	msgs*	whatis*
edit*	page*	whereis*
error*	print*	which*
ex*	printenv*	whoami*
expand*	prmail*	whodo*
fmt*	reset*	xstr*
fold*	script*	yes*
from*	soelim*	

/dev

MAKEDEV	*lp?	pty[pqrs]?
MAKEDEV.local	makedev.sh	tty
console	*.mt*	ttyNN
cua0	null	tty[pqrs][0-f]
cul0		

This page intentionally left blank.

REFERENCES

[1] Department of Defense Trusted Computer System Evaluation Criteria, December 1985, DoD 5200.28-STD.

[2] Department of Defense Password Management Guideline, 12 April 1985, CSC-STD-002-85.

[3] System Administrator's Guide for Gould UTX/32S, Release 1.0, November 1986, System Administration Manuals, Volume 3 of Gould UTX/32S Document Set, Release 1.0, March 1986, Volume Order Number 323-005233-000

[4] System Administrator's Reference Manual for Gould UTX/32S, Release 1.0, March 1986, System Administration Manuals, Volume 3 of Gould UTX/32S Document Set, Release 1.0, March 1986, Volume Order Number 323-005233-000

[5] Installation Guide for Gould UTX/32S, Release 1.0, November 1986, Volume 1 of Gould UTX/32S Document Set, Volume Order Number 323-005231-000

[6] Software Release Notes for Gould UTX/32S, Release 1.0, April 1986, System Administration Manuals, Volume 3 of Gould UTX/32S Document Set, Release 1.0, March 1986, Volume Order Number 323-005233-000

[7] Using Gould UTX/32S, Release 1.0, November 1986, Volume 1 of Gould UTX/32S Document Set, Volume Order Number 323-005231-000

[8] User's Reference Manual for Gould UTX/32S Volume 1, Release 1.0, March 1986, Volume Order Number 323-005231-000

[9] User's Reference Manual for Gould UTX/32S Volume 2, Release 1.0, March 1986, Volume Order Number 323-005232-000

[10] Gould V6 and V9 CPU Reference Manual, March 1985, Publication Order Number 301-003410-000

[11] Gould Virtual Memory Diagnostic Facility, Release 1.1, User's Manual, November 1984, Publication Order Number 326-005640-100

[12] Guide to the Security Test Suite for Gould UTX/32S, Release 1.0, Version 1, August 8, 1986

[13] Security Policy Model for Gould UTX/32S, Release 1.0, July 11, 1986

[14] UTX/32S 1.0 Security Test Suite SOURCE LISTING, July 18, 1986

[15] Program Maintenance Manual for Gould UTX/32S, Release 1.0, July 11, 1986

[16] 4.2BSD and UTX-32 Differences Study for Gould UTX/32S, Release 1.0, December 1986

[17] Traps and Interrupts for Gould UTX/32S, Release 1.0, December 1986

[18] Memory Management for Gould UTX/32S, Release 1.0, December 1986

[19] Object Reuse Study for Gould UTX/32S, Release 1.0, December 1986

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) CSC-EPL-86/007	5. MONITORING ORGANIZATION REPORT NUMBER(S) S 228,373
6a. NAME OF PERFORMING ORGANIZATION National Computer Security Center 6b. OFFICE SYMBOL (If applicable) C1	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State and ZIP Code) 9800 Savage Road Ft. George G. Meade, MD 20755	7b. ADDRESS (City, State and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State and ZIP Code)	10. SOURCE OF FUNDING NOS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT NO.
11. TITLE (Include Security Classification) Final Rpt. Gould UTX/32S	
12. PERSONAL AUTHOR(S) Baker, Dixie B. (Aeorspace Corporation); LaFountain, Steven M. et al	
13a. TYPE OF REPORT FINAL 13b. TIME COVERED FROM _____ TO _____	14. DATE OF REPORT (Yr., Mo., Day) 861231 15. PAGE COUNT 62
16. SUPPLEMENTARY NOTATION	
17. COSATI CODES FIELD / GROUP / SUB. GR.	18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) Trusted Computer System Evaluation Criteria, Gould, UTX/32S, UNIX, C2, EPL, NCSC

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

The security protection provided by release 1.0 of the Gould UTX/32S operating system running on a Gould PowerNode processor has been evaluated by the National Computer Security Center (NCSC). The NCSC evaluation team has determined that the highest clas at which UTX/32S satisfies all the specified requirements of the Criteria is class C2 and therefore, Release 1.0 of UTX/32S has been assigned a class C2 rating.

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

cont'd

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

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

19 cont'd.

The UTX/32S operating system is a general-purpose, time-sharing system that provides two basic security mechanisms. The restriced environment provides separation between administrative and normal users and files. The protection bit mechanism provides discretionary access control.