AEROSPACE REPORT NO.
ATR-88(3770-29)-1

(2)

Guidelines for Audit Log Mechanisms in Secure Computer Systems

R. L. BROWN
Computer Security Office
Systems and Computer Engineering Division
The Aerospace Corporation
El Segundo, CA 90245

12 November 1987

Prepared for

NATIONAL COMPUTER SECURITY CENTER
Ft. George G. Meade, MD 20755-6000

Engineering Group

THE AEROSPACE CORPORATION

Aerospace Report No.
ATR-88(3770-29)-1

GUIDELINES FOR AUDIT LOG MECHANISMS IN SECURE COMPUTER SYSTEMS

Prepared by
R. Leonard Brown
Computer Security Office
Systems and Computer Engineering Division

12 November 1987

Engineering Group
THE AEROSPACE CORPORATION
El Segundo, California 90245

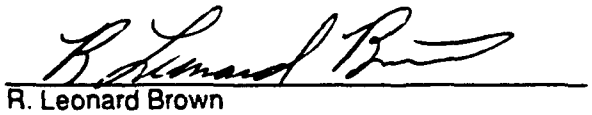
Prepared for
National Computer Security Center
Ft. George Meade, MD 20755-6000

APPROVED FOR PUBLIC RELEASE;
DISTRIBUTION UNLIMITED

Aerospace Report No.
ATR-88(3770-29)-1

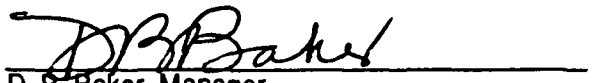
Guidelines for Audit Log Mechanisms in Secure Computer Systems

Prepared



R. Leonard Brown

Approved



D. B. Baker, Manager
Secure Systems Section

D. D. Downs, Director
Computer Security Office

ABSTRACT

This report summarized the process of creating an audit trail for the operation of a trusted computer system, and relates that process to the requirements of the Trusted Computer System Evaluation Criteria of the National Computer Security Center. The features required at each level of trust within the evaluation criteria are listed, and corresponding assurance requirements are presented. Implementation options that are available and recommended are discussed. The effect of trusted computer system structure on the audit mechanism implementation and on the choice of audit reduction tools is discussed. Anticipated near term technology advances are described.

CONTENTS

1. OVERVIEW

In August 1983 the Department of Defense Computer Security Center (now called the National Computer Security Center, NCSC) issued the *Trusted Computer System Evaluation Criteria*[1] (*TCSEC*), which specifies seven levels of effectiveness of security controls for computer systems that use shared resources. These criteria were created to encourage ADP system vendors to provide systems which conform to Department of Defense security requirements, and to allow DoD programs to select off-the-shelf ADP systems for the processing, storage and retrieval of sensitive or classified information. The NCSC uses these criteria to evaluate ADP systems submitted to it for evaluation by the system vendors. Once a system has been formally evaluated by a team that includes NCSC personnel as well as consultants from Federal Contract Research Centers (FCRC), a description of the hardware and software that make up the ADP system, together with its rating, is added to an Evaluated Products List (EPL) maintained by the NCSC. In addition, a Final Evaluation Report is published giving details of how the team arrived at its rating.

Since the original publication of the *TCSEC*, the NCSC has seen its mandate extended to providing technical guidance for the purchase of all computers by the Federal government. As of this writing, a number of systems have been evaluated and have been, or are about to be, added to the EPL. In order to gain from the experience of the teams who performed the first evaluations, and to provide additional guidance to vendors on what constitutes acceptable levels of protection, guidelines are being published that address various aspects of secure computer systems. The *TCSEC* itself contains initial guidelines concerning covert channels, mandatory access control configuration, and security testing. Additional guidelines have been prepared concerning the required levels of protection for various security environments[2], and use of passwords for authentication of users to the computer systems[3]. The password guideline has been issued by the Center, and the environmental guideline is under publication review. This report was prepared under contract to the NCSC for use in preparing a guideline for the audit mechanism required at all levels at or above C2 of the *TCSEC*.

1.1. Security Policy and the Audit Mechanism

The Department of Defense *ADP Security Manual*[4] provides the underlying policy that resulted in the audit log criteria of the *TCSEC*. The manual requires that all personnel having access to classified information be individually identifiable, which gives rise to the requirement for individual accountability. In addition, procedures must exist for detecting the improper disclosure of classified material through system malfunction or personnel action. Further, the Manual requires the effective use and disposition of system audit records, including records of security violations or security related malfunctions. The DoD *Industrial Security Manual*[5] describes audit trails and lists three areas of concern:

1. preparation of input data and dissemination of output data,
2. activity involved within an ADP environment (e.g., ADP support personnel modification of security controls),
3. internal machine activity (e.g., access to memory segments controlled by a protected process).

Further requirements are that the system security supervisor review recorded audit trails at least weekly, and that the audit trails be kept for one inspection cycle. These government policies led to the audit criteria in the *TCSEC* which a computer system must meet for the level of protection required for processing data within the accepted bounds given by the environmental guidelines[2]. Of course, the DAA is the final arbiter in these matters.

1.2. Definitions

In this section, several terms that are used extensively in this report are defined.

A *Security Policy Model*, such as the Bell and LaPadula model[6], is an informal presentation of a formal security policy model. Such a formal model is a mathematically precise statement of the set of laws, rules, and practices that regulate how an organization manages, protects, and distributes sensitive information. In addition, the discretionary security access model, which allows or prevents particular subjects access to named objects, may be included in the policy model. Very little work has been done on formalizing the statement of discretionary access control. No formal model is required at some levels of trust, but a formal model is required for high level of trust systems. A guideline on formal models is in preparation.

A *Trusted Computing Base* (TCB) consists of the totality of protection mechanisms within a computer system which, in combination, enforce the security policy, as described by the security policy model.

A *subject* is either an authorized user of the ADP system, including the operator and system administrators, or a process which is running on his behalf and has the same access privileges as the authorized user.

An *object* is an entity that contains or receives information. Depending on the system, objects may include records, blocks, pages, segments, directories, directory trees, mailboxes, and programs in both source and executable form. If it can be specified by name by a subject, then a bit, byte, word, field, processor, communication line, clock or network node could also be an object. Subjects may be treated as objects. For example, if a subject is allowed to spawn child processes, those processes may be treated as objects.

The word *audit* has various meanings depending on the context in which it is used. Generally, a computer audit involves an inspection of the *procedures* surrounding the administration of the computer center. Of particular interest are billing practices where security considerations are limited to the accuracy and inviolability of the billing log. Less interest is taken in the security of the actual records stored in files and data bases. An inspection of the literature under the conjunction of the keywords *audit* and *security* turns up many papers such as that of van den Berg and Leenars[7] which address such an administrative audit, but almost none that address the security of data. The Department of Defense

usage of the term audit is closer to the dictionary term, which defines audit as "to examine and check." In particular, if the TCB of an ADP system mediates every access of a protected object by a subject, then examining each of these transactions is possible.

The TCB must be able to *log*, or record, any auditable action in addition to examining each access. Further, certain actions must always be logged. This guideline will discuss mechanisms for recording and processing the audit log as required by the *TCSEC*. The major concern in logging auditable actions is that the sheer volume of material logged might make the detection of a security violation impossible. The result would be the inability to discover when an illegal disclosure has occurred. This would be a violation of the requirements of the DoD ADP Security Manual[4].

1.3. Outline of Report

In the next section, each requirement from the *TCSEC* is stated in full, followed by a discussion of its effect on the audit log mechanism. The format from the Requirement Directory of the *TCSEC* is given, where only NEW or ADDed requirements are shown and discussed. The section is divided into two parts: audit features and assurance requirements. This is to insure that the distinction between the two types of requirements is maintained. One reason for maintaining this distinction is that the implementation of features tends to require different skills than some of the assurance requirements. Features tend to be programming intensive, involving designing and maintaining large data structures. Assurance requirements tend to involve management skills, such as for configuration management, and mathematical skills, such as for formal specification and verification.

Section 3 discusses those requirements for which the implementation is not obvious, or for which several implementations are possible.

Section 4 presents several alternative designs for reduction tools, which allow the system administrator to search the audit log for particular profiles of usage which might indicate an attempt to disclose protected information. Such tools are modeled on current technology query languages based on data base management systems, but have some features of a report generator system as well.

The final section lists near term advances that are possible by combining current technology in as yet untried ways. Obviously, breakthroughs in either hardware or software technology cannot be predicted, but would be welcome.

2. THE TRUSTED COMPUTER SYSTEM EVALUATION CRITERIA

The *Trusted Computer System Evaluation Criteria*[1] provides for seven levels of effectiveness of security controls divided into four divisions. The requirements of each level include those of all lower levels. Throughout the levels of the *TCSEC*, increasing effectiveness in protecting the system is achieved through a combination of protection features and assurance requirements. In general, a security feature is a required part of the TCB which enhances the ability of the system administrators to protect the information stored in the ADP system. The *TCSEC* attempts to require such features without specifying their specific form of implementation, thus allowing vendors the widest latitude in adapting their particular hardware innovations and software design philosophies to the needs of the user who requires data protection. Assurance requirements provide both the NCSC evaluators and the system security administrator confidence that the data are being protected, and that any unauthorized disclosure of protected data can be detected and dealt with appropriately.

Division D systems provide minimal protection and are suitable only for use in a dedicated security mode. Division C systems provide discretionary access controls, by which access of a protected object may be permitted or denied based on the identity of the subject seeking access. Class C1 does not require individual accountability, whereas a Class C2 system must be able to individually identify each user, and both allow and deny access to an object by a particular named user. C2 is the first level at which audit capabilities are required, since this is the lowest class at which individual accountability is possible. Mandatory access control, which includes security classification, is not addressed within Division C.

Division B systems enforce mandatory security, which is exemplified by the military model of security which requires both subjects and objects to be characterized by a security level. Each subject must be operating at a particular security clearance level before being allowed access to a protected object; this protection is in addition to the discretionary access control enforced by lower level systems. In addition to security levels, an additional restriction involving category sets can be invoked. If a subject does not have access to a category in which an object is placed, then the subject is denied access to that object, even if its level has the correct relation to the security level of the object.

This type of organization is generally described as a *lattice model* of security. A lattice is a finite set together with a partial ordering on its elements such that for every pair of elements there is a least upper bound and a greatest lower bound. For example, in the military system of sensitivity levels and compartments, a subject with a specific clearance (sensitivity level) and compartment set *dominates* an object with a specific classification (sensitivity level) and compartment set if the subject's clearance is no less than the object's classification, and the object's compartment set is a subset of that of the subject. Thus, the partial ordering function is *dominates*. In most security policy models such as that of Bell and LaPadula[6], the subject would then be allowed read access to the data stored in any object it dominates.

The three classes of Division B provide increasingly greater effectiveness of protection within the

mandatory access control framework. As in Division C, this increasing effectiveness is achieved by two means: additional protection features, and additional assurance requirements. A B1 level system has no assurance requirements beyond those of a C2 system, except for requirements related to the introduction of mandatory security. Starting with level B2 systems, increasing assurance requirements are introduced. These are generally software engineering requirements that are applied during the development of the system.

Division A systems add more assurance requirements by mandating the use of state of the art design specification and verification languages to *prove* that the TCB, as specified to the programmers who write the final code, conforms to the formal model of the security policy and the formal top level specification of the system design.

In the following two sections, audit features are described by class, and then assurance requirements on the audit mechanism are analyzed by class.

2.1. Audit Features

The new audit criteria for each increasing trust level are described in the following sections. Each criterion number is shown in square braces to the right of its descriptive name. The first level of the *TCSEC* that requires any audit features is the C2 level.

2.1.1. The Audit Process

Discretionary Access Control [2.2.1.1]

C2: CHANGE: 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... ADD: These access controls shall be capable of including or excluding access to the granularity of a single user.

Audit [2.2.2.2]

C2: NEW: The TCB shall be able to create [and] maintain ... an audit trail of accesses to the objects it protects. ... 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, and actions taken by computer operators and system administrators and/or system security officers. 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.

Trusted Facility Manual [2.2.4.2]

C2: ADD: The procedures for examining and maintaining the audit files as well as the detailed audit record structure for each type of audit event shall be given.

The definition of audit, "to examine and check," implies a two stage application of the TCB to each security related event. Of course, if all auditable events are recorded, the second stage, which consists of

checking whether to record the event, is unnecessary. Starting at the C1 level, the TCB is already required to mediate every access attempt by a "named user" to a named object. The named user may correspond to a subject, but since there is no individual accountability required at the C1 level, it could also be a program only loosely identified with a group of users. The TCB is also required to authenticate each named user who attempts to logon to the system; this accounts for the remainder of the events mentioned in the requirements. The second half of the definition of audit requires a check to determine whether the event should be recorded in an audit log. It is possible, but not required, to keep separate logs for ease of processing or greater security. For example, at higher levels all actions of the system security administrator are auditable, so it is possible to keep a separate log of these events when they are being logged.

This breakdown of the audit process into two components makes the system design team's task in designing the TCB straightforward: for every type of event which the TCB mediates, they must decide whether the event should be selectable (some security related events, such as a housekeeping type program within the TCB accessing a secure file, need not be recorded). If the event should be selectable, then the system must include code to determine whether the system security administrator has selected that event to be recorded. *The requirements in the TCSEC do not include the recording of any audit data, only the ability to record such data.* If the administrator does not enable recording of a class of events, then those events will not be recorded.

The actual design of the audit mechanism is more straightforward, but must be efficiently implemented, since the enabling of all audit events could result in a significant degradation in processor speed. Just the act of determining whether an event mediated by the TCB is subject to audit, and whether the event has been marked for logging, could take significant processor resources. Some suggested implementation details appear in Section 3.

An additional problem arises from the requirement that the administrator may specify different logging requirements for each user. A system with many users may pay a large price to meet the audit requirement, either in random access memory or in time lost checking an externally stored list as each auditable security related event is encountered. An alternative audit mechanism that avoids the checking mechanism is simply to record every security related event, as is done in the CDC NOS dayfile. This generates a large quantity of data which requires sufficient storage and a sophisticated reduction tool, as described below.

Another consideration is that individuals are now distinguishable, and the discretionary access control mechanism allows the creation of groups of users, all members of which may be given specific access rights to an object. Creation, addition to, and removal of users of such a group should be auditable.

2.1.2. Recording Medium

As will be seen in the assurance section below, the audit trail, or audit log, must be protected. Only "those who are authorized for audit data" may see the data. The system administrator responsible for security and only a few others should be included. At the *TCSEC* levels below B3, one or more users are identified to the system as system administrators. At the B3 and A1 levels, a particular system security administrator must be appointed, and action of assuming this role must be auditable. In the following, the term *administrator* refers to the appropriate system administrator or system security administrator, depending on the level being discussed. One reason for the constraint on access to the audit log is that data from failed authentication attempts may include the text of the failed attempt, and legitimate users who type in their password incorrectly would record in the audit log most of a legitimate password. The Password Management Guideline[3] recommends that passwords, including attempted passwords, not be included in the audit trail. Note that in some systems, a user may inadvertently enter the correct password when the system expects to see the user's identification. In this case, if the system developers have chosen to record invalid user names, the audit record will contain a valid password. The choice of whether to record invalid user names is left to the vendor[8].

A second reason for protecting the audit data is that the aggregation of data in the audit log, even if failed password attempts are not recorded, represents highly sensitive data. This is often referred to as *aggregation* of information. While each piece of information recorded in the log may not provide much information concerning the system and its users, a careful reading of the totality of the data may allow an intruder to infer information that would be considered highly sensitive. Therefore, the audit log must be protected at the highest level defined on the system.

A second consideration that must be addressed is availability of the audit log data. There is no requirement that the audit log data be immediately accessible to the system administrator or system security officer, and the major hardware implementations discussed in Section 3 will not allow for immediate access to the data

2.1.3. Reduction Tools

Although the *TCSEC* requirements do not specifically include software to reduce the mass of data that could be acquired by the audit mechanism, the existence of such reduction tools is implicit in the wording of the requirement. Even if the TCB can be configured to record only selected security related events, the worst case of every event being recorded can occur. Since "The ADP system administrator shall be able to selectively audit the actions of any one or more users based on individual identity," where the term *audit* here refers to a person's action, not that of a machine, the worst case scenario would be that the administrator suspects an attempt will be made to disclose protected information, orders all security events be audited, then sits down with the audit record and personally inspects (audits) individual user's audit messages in the order of least trusted user first.

Since a considerable amount of data is required to be recorded for each event, both storage and

computational efficiency suggest that most of the data be encoded. See Section 4 for a discussion of how reduction tools are used to search the audit logs.

An audit reduction tool is a combination of query language and data base manager. A report generator must be available to expand the encoded fields to human readable form, to format reports generated from the raw audit records, and to perform fairly complex accesses of data across several fields and many records. For example, a typical request might be "Display all failed file access attempts made by user1, user2, and user3 between 0600 and 1200 today." This could be used to test the hypothesis that at least two of these individuals were passing data through a timing channel to each other. Clearly, such information could also be obtained by a series of queries, each producing a file of audit records. Then other types of software such as an editor could be used to extract common records. This type of system would satisfy the requirement within the *TCSEC*, but would be more difficult to use than a query language based system.

The system designer should consider very early whether new software should be written to perform the reduction function, or whether existing software such as a data base manager could be modified to perform it. It is possible that the reduction tools have access to files that should be protected at the highest level within the machine. If the information compiled from those files might be stored at a lower level of protection, then the audit reduction tool must be considered a trusted process. In any case, the reduction tools are part of the trusted software of the TCB.

2.1.4. Significance of the Audit Mechanism

For most of the elements of the *TCSEC*, as a system's evaluation level increases, so do the number and sophistication of its features. In general, new features are only added to the audit log mechanism in response to additional features in other areas. However, at the C2 level the audit trail is the major defense against unauthorized disclosure. Neither the modular organization of the TCB, nor the configuration management during the design process, both required at higher levels, are required at C2. During the limited testing phase required at the C2 level, a full audit record should be maintained. Then the evaluators can verify the functioning of the audit mechanism. The administrator must feel confident in using the audit data to try to determine whether successful or failed attempts to compromise the data on the machine have been made. This is addressed further in the section on assurance requirements.

2.1.5. Significance of the Trusted Facility Manual

Since the administrator will rely so heavily on the audit data at the C2 and B1 levels, a well written Trusted Facility Manual is at least as important as it is at higher levels of trust. The recommended approach is to include a chapter on the use of the audit reduction tools, relegating to an appendix the actual record structure of the audit events logged. The user's manual would refer to this appendix when it explains how the reduction tools work. The Trusted Facility Manual should also explain in detail how the security administrator uses the commands to enable and disable logging of events by user, if security

events can be selectively logged on the system.

2.1.6. Log Record Requirements

At Division B, the introduction of classification levels, and the requirement for security marking of exported data, add additional features to the TCB which must be audited. The *TCSEC* requires:

Audit [3.1.2.2]

B1: CHANGE: For events that introduce an object into a user's address space and for object deletion events the audit record shall include the name of the object **and the object's security level**. The ADP system administrator shall be able to selectively audit the actions of any one or more users based on individual identity **and/or object security level**. ADD: The TCB shall also be able to audit any override of human-readable output markings.

Exportation of Labeled Information [3.1.1.3.2]

B1: NEW: The TCB shall designate each communication channel and I/O device as either single-level or multilevel. Any change in this designation shall be done manually and shall be auditable by the TCB. The TCB shall maintain and be able to audit any change in the current security level associated with a single-level communication channel or I/O device.

Clearly, the structure of the audit record includes the security level of objects that subjects attempt to access. This was not a requirement at the C2 level; however, to maintain a consistent structure for audit records, it is possible that C2 audit records also contain this field. This would allow a C2 system to later evolve to a B1 system, since these two levels within the *TCSEC* differ mainly in features, not in assurance level. The most common mandatory security model, that of Bell and LaPadula[6], requires that the clearance level at which a user is operating (assumed to be dominated by the user's maximum level) dominate that of any object to which he seeks read access, and be dominated by any object to which he seeks to write or append. Therefore, the TCB of a B or A Division system must examine these levels at every access attempt. Both unsuccessful and successful attempts can then be recorded, along with the level of the object. Although it is not required, the user's maximum allowable level could also be included in the initial login record. Including these levels would enable the security administrator who inspects the audit log to determine whether an unsuccessful access attempt was an attempt to subvert the system, or attributable to user error or unfamiliarity with the system.

An entirely new requirement on the TCB software arises from the requirement that any attempted access of objects at a particular security level be auditable. While this might be restricted to one or a few users, it is possible that the security administrator require all user accesses be logged. The implementation of this feature is discussed in Section 3.

Exportation to Single-Level Devices [3.1.1.3.2.2]

B1: NEW: Single-level I/O devices and single-level communication channels are not required to maintain the sensitivity labels of the information they process. However, the TCB shall include a mechanism by which the TCB and an authorized user reliably communicate to designate the single security level of information imported or exported via single-level communication channels or I/O devices.

An "authorized user" is a user who has been identified to the system through its identification and

authentication mechanism, and whose hierarchical class and non-hierarchical category permissions dominate the security level of the information to be sent via the single-level channels or devices.

2.1.7. Human-Readable Output Markings

The requirement that the TCB shall be able to audit allowable override of human-readable hardcopy output markings by authorized users does not imply that the ability to override such markings be part of the TCB. Here allowable implies only inclusion or exclusion of the markings. Also, the actual value of the classification and categories printed should not be changed. Other changes, such as change of placement or change of default type style and size, may be user or administrator specified, either permanently or for the duration of one process. The banner page marking is not subject to being overridden, since it can be stripped off manually. The top and bottom page marking override is useful if the computer will be used to process special forms, which would be typical of lower level secure systems. One example of this is printing continuous form checks on a system shared by payroll and other departments. There should be an option that allows the administrator to turn off the ability of these authorized users to override top and bottom page markings when the system is being used only for secure applications that do not require special forms. For example, an embedded computer system controlling a military application would not need such special forms.

2.1.8. Single and Multi-level Devices

At Division B, systems are multi-level, and some (at the B1 level) or all (at B2 and above) input and output devices must come under control of the TCB. However, any devices not under control of the TCB should not be addressable by the TCB. An input/output device is defined as one capable of transferring information from/to personnel or systems (ADP or otherwise) outside the control of the TCB. This definition excludes system disks, but includes disks with removable disk packs, unless physical security is required as part of the ADP system to control the removable disk packs. Also included are communications channels to terminals, remote job entry stations, and networks. The distinction between single level and multi-level devices is one of time dependence: a single level device always processes data as if it were in communication with a user operating at that clearance level (lattice model of mandatory access control), whereas a multi-level device is treated as if the system were communicating with a user who could execute a request to change his operating level. As a result, the TCB must maintain a current security label and device maximum and minimum labels for each multi-level device. The latter is normally the case with terminal communication lines. Remote job entry stations are usually in a secured area, but the level of data to or from them depends on the physical security of their environment. Devices such as line printers and tape drives are normally multi-level if they are within the same physical environment as the TCB, but could be treated as single level at the direction of the system security administrator.

The auditing requirements treat the change of a device from single to multi level, or back, or the change of a single level device from one level to another, the same way it treats any actions of the

security administrator. If security administrator audit logs are kept separate from user audit logs, then this action belongs with the former.

Sending an object's contents to an output device is analogous to sending data to a terminal, and this should be logged with user interaction logs. Similarly, receiving data from a tape drive or remote job entry station is analogous to receiving data from a terminal, and should be logged similarly.

2.1.9. Security Administrator Role

At the B2 level, separate operator and system administrator functions are required. At the B3 level the system security administrator function must be identified. The specific functions relating to trusted facility management and audit must be determined by the system design team. A sample listing of the various personnel divisions might include the functions of security administrator, account administrator, trusted system programmer, secure operator, and operator[9]. Although this is not the only way to separate the various security related functions from those not security related, it is an interpretation that conforms to the requirements in the *TCSEC*.

Trusted Facility Management [3.3.3.1.4]

> B3: ADD: ... The ADP system administrative personnel shall only be able to perform security administrator functions after taking a distinct auditable action to assume the security administrator role on the ADP system.

At the B3 level, the additional feature is added that the security administrator must be identified to the system before changing any security features, including audit log instructions. At lower levels, anyone who has access to the operator's console, or any user whose account identifier is on a short list of security administrators, could perform these security relevant actions. The action of assuming the security administrator role must now be included in the audit log, if that test is enabled.

2.2. Assurance Requirements

While the security features allow the system security administrator to exert some control over user access to objects, the *TCSEC* assurance requirements specify how effective this control will be. At lower levels, simple checks of access control lists are easily subverted. In fact, although the system architecture requirement of the *TCSEC* specifies that the TCB maintain a domain for its own execution that protects itself from tampering with its code or data structures, even at the C1 level, the simple security testing requirements at the lower levels do not assure that tampering could not be performed by a persistent user. The purpose of the assurance requirements is to convince the vendor, the evaluators, and ultimately the users of a secure computer system that the security features of a system will protect data from disclosure, either by working as specified or by informing the system administrator of imminent threats of disclosure.

2.2.1. Protection of Audit Log

Audit [2.2.2.2]

C2: NEW: The TCB shall be able to ... 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.

Security Testing

C2: ADD: Testing shall also include a search for obvious flaws that would ... permit unauthorized access to the audit ... data.

In addition to examining each security related event, and checking to see if it must be logged, the TCB must provide some assurance that the audit log will still be there when the administrator tries to use it. This can be done using some combination of the system's own security features and hardware devices. If the audit records are protected by Discretionary Access Control mechanisms, then they are as vulnerable as any other records in the system. If a penetrator takes over the TCB kernel, these records may be rewritten to hide the successful penetration, so some type of offline storage or hardware protection is recommended for systems at the lower assurance levels (C2 /B1). In general, the routines for accessing the audit log, if it is on a system disk, should be among the most protected software within the TCB. Only the system security administrator should be allowed to run these routines, and they should be run in an isolated fashion in part of the protected domain of the TCB. At the C2 level examination of audit log data is the primary means of detection of actual or attempted disclosure of protected information. If the log itself is as easy to subvert as other data, a penetrator will try, and possibly succeed, in removing evidence of his access to sensitive data by altering the audit log.

The testing requirement recognizes the vulnerability of the audit log, and requires that an attempt be made to find obvious flaws that would allow one to read or modify the log. This is a stronger testing requirement than at the C1 level, where the system is only required to be exercised to see that each command functions as documented.

2.2.2. Audit of the TCB's Execution Domain

System Architecture [2.2.3.1.1]

C2: ADD: The TCB shall isolate the resources to be protected so that they are subject to the ... auditing requirement.

At the C1 level, the TCB is required to have a specific execution domain for its code and data structures. At the C2 level, this area of storage is required to be identifiable, and any attempt to read from or write to this area is subject to auditing. Since the code and data segments, pages, or other memory partitions of protected elements are treated as objects by the TCB, this auditing is not technically difficult. It is important to log such access attempts since they may indicate an attempt to modify the protection mechanism of the system. At the C2 level, where the audit log is the primary means of detection, the initial target of such modification could be the audit mechanism's code or its log data. Although it is not required, it is recommended that the audit mechanism always log such attempts, and use a special

logging device, such as the operator's console, to record such attempts. Then, even if the audit mechanism were successfully compromised, a hardcopy printout of the initial attempt would exist.

2.2.3. Neutralizing Flaws Through Auditing

Security Testing [3.1.3.2.1]

B1: NEW: The security mechanisms of the ADP system shall be tested and found to work as claimed in the system documentation. ... All discovered flaws shall be removed or neutralized and the TCB retested to demonstrate that they have been eliminated and that new flaws have not been introduced.

As noted above, the transition from C2 to B1 levels entails an increase in functionality of the TCB, with little significant increase in assurance. One exception is an increased requirement for testing, since at B1 level mandatory access control in introduced. The testing requirement involves checking the functionality of these mandatory controls. However, flaws found in testing may be *neutralized* in any of a number of ways. One way available to the system designer is to audit all uses of the mechanism in which the flaw is found, and to log such events. Naturally, it is better to have removed the flaw, but the *TCSEC* does not require it. Starting at level B2, all flaws must be removed.

2.2.4. Covert Storage Channels

Audit [3.2.2.2]

B2: ADD: The TCB shall be able to audit the identified events that may be used in the exploitation of covert storage channels.

Design Documentation [3.2.4.4]

B2: ADD: ... All auditable events that may be used in the exploitation of known covert storage channels shall be identified. The bandwidths of known covert storage channels, the use of which is not detectable by the auditing mechanisms, shall be provided.

There is a considerable difference between the assurance requirements of level B1 and level B2. For the first time, the actual structure of the TCB is required to conform to certain standards. These standards make testing of the TCB easier. In addition, since all of the code and data structures that are security relevant are isolated into distinct modules, it is possible to identify any storage areas that are accessible to more than one user. If these users are at different classification levels, yet one can read data into such an area and the other write data into it, then a covert storage channel exists. The system designers must eliminate all such channels that serve no purpose, but it is possible that the system will not function well, or at all, if certain channels are removed. Therefore, the occurrence of events that signal the possible use of such channels must be identified. Normally, these are already security relevant events as discussed in 2.1.1. Also, the bandwidth of such channels needs to be reduced, if possible, especially if technical reasons prevent their being audited.

2.2.5. Real Time Notification

Audit [3.3.2.2]

> B3: ADD: The TCB shall contain a mechanism that is able to monitor the occurrence or accumulation of security auditable events that may indicate an imminent violation of security policy. This mechanism shall be able to immediately notify the security administrator when thresholds are exceeded.

At the B3 level, the combination of system architecture requirements and covert channel analysis enables the designer to identify sequences of events that might indicate an illegal access attempt *before* it is accomplished. For example, a series of opening and closing of files that can be read by a subject operating at secret level and written by a subject at unclassified level might indicate the use of a timing channel. The actual transfer of data through this channel occurs when both subjects use a clock to control their query to the system of whether the file is open or closed, since read or write access both require an open file. If the formal security model allows subjects to write up and read down, there is no way to preclude this sort of channel; the *TCSEC* requires only that its existence be identified and that there exist a mechanism that signals events that might indicate that the channel is in use. There is no requirement to simply look for suspicious activity other than that identified during the design phase. This is not restricted to timing and storage channels, but would include other events such as repeated invalid login attempts. See Section 3 for a description of the operation of the real time notification mechanism. See the Appendix for examples of such events.

2.2.6. Modular Organization and Its Impact on Audit

System Architecture [3.3.3.1.1]

> B3: ADD: The TCB shall be designed and structured to use a complete, conceptually simple protection mechanism with precisely defined semantics. ... Significant system engineering shall be directed toward minimizing the complexity of the TCB and excluding from the TCB modules that are not protection-critical.

At the B3 level, it is possible that the required modular organization of the TCB will both make the decisions on what to audit easier, and actually reduce the total number of events that must be audited. The Descriptive Top Level Specification (DTLS) required at level B2 must describe the TCB "in terms of exceptions, error messages, and effects." This has been interpreted as a requirement that the TCB must structurally conform to the DTLS, in addition to its being "internally structured into well-defined largely independent modules." At the B3 level, the TCB must have a conceptually simple protection mechanism with precisely defined semantics.

Significant work has been done on how to describe the allowable transitions of data within a secure system. There are two major models of such systems[10]: the finite state machine model which describes access controls in terms of states and state transitions, and information flow models that add an information flow policy on top of the state transitions such that the only allowable transitions are those represented by an element of the policy. The most often cited example of the former is found in Bell and LaPadula[6], and the latter is in Denning[11]. Both are based on the lattice model of security, described above. More recent developments have included data flow languages and concepts that do not require states and state transitions. A method based on seals and signatures exchanged among processes[12]

could be implemented with its processes corresponding to elements of the TCB interacting with user processes (subjects). A guideline on descriptive and formal top level specifications is in preparation.

Whatever model and corresponding DTLS is chosen, the basic actions (state transitions or data flow transactions) correspond to the auditable actions of the system. Therefore, by enforcing a specific protocol for invoking one of these actions which includes a call to the audit mechanism, the actual audit process is simplified. Since the TCB must be engineered to minimize its complexity and to include only protection critical modules, the total number of places in the code which require the invocation of the audit mechanism can be minimized. Then graph theoretic techniques can be used to further reduce the total number of points within the state transition graph, or equivalent dataflow graph, so that only uniquely identifiable vertices need be audited since it would be possible to determine all intervening transitions that led to that unique vertex. This would reduce the total number of events that would have to appear on the audit log. The full audit path could still be reconstructed using the audit log data and the audit reduction (or in this case expansion) tools.

2.2.7. Use of Formal Methods

Covert Channel Analysis [4.1.3.1.3]

A1: ADD: Formal methods shall be used in the analysis.

Design Specification and Verification [4.1.3.2.2]

A1: ADD: A formal top-level specification (FTLS) of the TCB shall be maintained that accurately describes the TCB in terms of exceptions, error messages, and effects.

At the A1 level, all elements of the TCB including the Security Kernel and trusted processes must be formally described using one of the specifications languages such as Gypsy[13], Ina Jo[14], or Affirm[15]. For example, the first A1 Level system, SCOMP, had its kernel specified and verified using Special, and its trusted processes specified and verified using Gypsy. The formal methodologies, Gypsy Verification Environment (GVE) in the case of Gypsy, Formal Development Methodology (FDM) in the case of Ina Jo, all encourage a modular, top down specification of the system software. Therefore, at one level in this top down development the individual components should each correspond to at most a single auditable event. Then both the formal specification of auditable events and the actual coding of the audit mechanism are made an integral part of the system development cycle.

In addition to using verification techniques to prove the correspondence of the system's specified modules to the FTLS, the system designers will also use the lack of proof of certain of the theorems derived during the development process to find storage and covert timing channels. Some of these channels may be removed, but usually several channels, especially timing channels, will remain and the only defense against them is to audit for events that might signal their use.

3. IMPLEMENTATION OPTIONS AND RECOMMENDATIONS

The problem facing the designers of a new system is the coordination of the available hardware with appropriate software engineering technology so that the resulting system meets the security requirements, yet is also efficient enough to be a commercial product. Since both hardware architectures and the design philosophies embodied in the software architectures differ for each manufacturer, often for each line by each manufacturer, and even for system architectures implemented on the same hardware (e.g., VMS and Unix on the VAX computers), no single implementation can be recommended for any required feature from the *TCSEC*. Instead, the following discussion embodies the limited experience that the DoDCSC evaluators have in evaluating existing commercial products. Some recommendations are based on systems that have been evaluated. Also, some attempted implementations that were seen not to work have affected the recommendations. Some of the implementations have not been used in any commercial product but are based on concepts from the field of software engineering. All recommendations are non-proprietary, and are offered for consideration by the system designer.

3.1. The Audit Mechanism

One simple solution for implementation of the audit logging mechanism, especially for Division C and level B1 systems which are not required to be as highly modular as higher level systems, is to initiate a special system call immediately after any of the events in the list of auditable actions. Events which the TCB mediates but does not audit would simply not have this call in the code which handles that event. The called routine itself would maintain, in secure storage, a table of events which the system security administrator has enabled. The table itself could be simply implemented as a bit-array if the processor is capable of fast bit manipulation. The intent of having the array in memory is to maintain processor speed. If a single byte or whole word for each enabled event would result in faster execution of this system call, then that should be used instead. Also, since success and failure could be treated differently, this should be accounted for. If bit maps are used, there would be a success bit and a failure bit for each auditable event. If bytes are used, then integer values of 0, 1, 2 and 3 could be used, respectively, to indicate event recording off, record success only (unlikely but possible case), record failure only, or record any access attempt, successful or failed.

On a system which supports only a few users logged on at a time, and which uses bit-maps to represent logging requirements, one simply keeps one bit-map per logged on user, storing the bit-maps of other users in a file until needed. Clearly, when the identification mechanism detects the entry of the identity of a valid user, that user's bit-map would be read from this file before the authentication mechanism ran. In systems which use virtual memory, each user's bit map can be part of his virtual address space, always kept in the same virtual location but initialized with different data.

3.2. Selective Audit Logging

Recording all security-relevant events in the audit log may produce an unmanageable amount of data. Since the *TCSEC* requires that the administrator be able to view auditable events selectively, one way to accomplish this is to allow him to specify that only certain events will be recorded, thus satisfying the requirement and reducing the amount of data.

There are four basic selection criteria that can be used, singly or in combination:

1. Selection based on type of auditable event
2. Selection based on user identity
3. Selection based on object identity
4. Selection based on object properties (usually security level)

Not all of these need be implemented, nor all possible combinations. Only the administrator should be able to change the selection criteria, and it should not be possible for a non-privileged user to determine whether a particular operation is audited, except perhaps by measuring the duration of the operation. At a minimum, however, both the ability to audit based on user identity and on object security level must be available[16].

3.2.1. Selection Based on Type of Event

An audit mechanism records many different events, but could be implemented only to create messages for certain specified events. Requiring that the administrator select individually each possible type of audit message to be recorded is inconvenient. It is better to group the potential audit messages into classes (possibly overlapping), and allow the administrator to specify which classes of events are to be logged. Examples of audit message classes include:

- Successful accesses to any objects
- Unsuccessful attempts to access any objects
- Accesses to non-file objects (tape and disk mounts, use of communications lines, etc.)
- Attempts to execute invalid machine instructions
- Accesses that modify objects
- Modifications affecting accessibility of an object (such as changes in protection attributes, access control lists, etc.)
- Administrative actions: creating/deleting users, modifying attributes of I/O devices or communication lines, etc.
- Privileged operations: actions by security administrators that bypass normal protection mechanisms.
- Operator actions, if different from administrator actions
- Potential covert channel use events

Which classes might be implemented depends entirely on the architecture and capabilities of the

system. In general, the first class (successful accesses to objects) will generate the bulk of audit messages in any application, and is the most important over which to have control.

3.2.2. Selection Based on User Identity

The audit mechanism can select on user or group identity. This allows the administrator to consider some users more trusted than others. If this is combined with selection by event class, it is particularly useful. Some users (the untrusted or particularly critical ones) can be monitored for all events, and others monitored only for events that might indicate problems (such as unsuccessful access attempts).

Selection by user identity can also be used to implement "spot checks," so that the security administrator can periodically (without warning) monitor all the activities of a user. This, too, reduces the bulk of audit data, while still offering some assurance that a perpetrator will be caught eventually. For example, one could divide the user community into ten groups and monitor the activities of each group on one randomly selected day in each two-week period. This is a common form of auditing in the "paper world" of commercial operations.

3.2.3. Selection Based on Object Identity

If there are particular objects of critical importance (such as the audit data itself, or user registration data, or an important application data base), it is useful to be able to specify that all accesses to those objects be audited, regardless of whether the access would have been audited according to any other selection rules.

Similarly, if auditing is enabled for successful accesses, it is useful to be able to specify that accesses to certain objects not be audited, again regardless of other rules. These objects might include common system library programs or other files of no security-relevant interest.

If the system has a hierarchical file structure, it is most useful if these controls can be applied to entire subparts of the hierarchy rather than having to be applied to each object individually. In addition to the simple convenience aspects, this capability is important for handling the creation of new objects within a subpart of the hierarchy, and ensuring that they receive appropriate auditing.

3.2.4. Selection Based on Object Attributes

The most important object attribute in this respect is security level. The *TCSEC* requirement for selection of audit messages by security level can be satisfied by auditing those objects at (or above) specified security levels. Other attributes, including system dependent ones, are selectable by the administrator.

3.3. Audit Log Storage

While the audit log data could be stored on direct access devices such as disk files, these files must rely on the TCB to ensure that the data are not compromised. Since a principal use of audit log data is detecting compromise of the TCB, it is possible that this data could have been altered during a successful infiltration of the system. A possible alternative depending on performance characteristics is to use a dedicated device, such as a tape drive or even an optical disk writer. Added physical security, such as disabling the read head of the tape drive after the beginning of tape mark is found, and not including a read mechanism on the optical disk writer, would isolate the audit log. It could be read only after the medium was physically moved to a different device, perhaps connected to a different ADP system such as a microcomputer dedicated to audit processing. Not only could a penetrator be prevented from reading the audit log, but a successful penetrator could not cover his tracks by deleting the history of his access to protected data. The main objection to this is that the system security administrator does not have immediate access to the audit log data, but must wait for it to be moved to the reading mechanism. However, there is no requirement in the *TCSEC* for immediate access, and for a system designed to be run unattended for long periods of time this would be an appropriate design.

An important requirement on the physical medium is that it be able to record at the fastest rate the data could come from the ADP system. The worst case scenario would be heavy usage with all audit records enabled. Notice that the absolute highest rate of audit log generation might peak before maximum rate of (unlogged) file access, since the accessing and releasing of files would significantly slow the TCB down because of all the checks required. This worst case rate can be estimated by the system designers and a suitable physical device and output buffering scheme within the TCB can be chosen to allow for this rate.

In the rare event that the highest anticipated data rate is exceeded, there are several actions the TCB could take. The system could delay all other processes until subsequent audit records could be written. The message that audit data were lost could be written to an alternative site, such as the hardcopy console. The system could be automatically shut down. It could write a message into the log that audit records were lost. This last is the suggested implementation, provided the lost message event is shown to be a rare event, on the order of the frequency that the storage medium developed a serious flaw.

The capacity of the storage medium is also an important consideration. The medium should not have to be removed and archived so often that the operator would be tempted to ignore it. Frequency of removal depends on the anticipated usage of the system: a stand alone minicomputer used by a small group without a full time operator would probably need its audit log medium archived once a day. This function could be performed by the system security administrator. A larger machine with a full time operator, or possibly sharing an operator with other machines, could require archiving of the physical medium as often as once an hour. In either case, the TCB must be able to warn the responsible person some time before the medium is full, and to lock all further auditable accesses once the medium is full until it is replaced.

In case the responsible person does not act in time and it becomes impossible to log further events, there are three possibilities. The machine can stop processing and wait until the medium is replaced. The audit records could be stored temporarily in the regular file system, usually in a highly protected file on the main system disk. Notification should be sent to all users online that the audit log medium is full but that the machine will continue processing without recording audit data. Two or all three of these options should be available on the system, and should be set by the administrator. The third option is particularly dangerous since it signals any penetrators who have access to the system that their actions will not be audited; however, it is a valid alternative because the administrator who allows this option might know that there are no sensitive files or devices currently available.

In the case of not using a dedicated audit device, but instead using the machine's own file system to store audit data, the implied archival requirement[4] that the data be stored for one full inspection period still exists. Therefore, the audit log data must be dumped to an archive device, such as a tape, using the TCB's highest protective abilities. Once transferred, the archive medium is placed under physical security control.

3.4. Data Structure of Audit Log Record

In general, the format in which data is to be recorded in each audit record is specified in *TCSEC*. This format should be chosen to optimize data access from the physical medium used to record it. The basic record entries required are listed in the C2 level requirements, whereas at the B1 and higher levels the clearance level at which a subject is operating must also be recorded. This can be implemented by adding fields to the audit log bit map of each user, by a separate table for each level in the TCB which has a list of users whose access attempts at that level are to be logged, or by logging all access attempts and using reduction tools to separate access attempts by level.

3.5. Audit Log Protection

The best approach to protecting the audit log is to send the data immediately to a dedicated device, such as a magnetic tape drive. At the least, the operator might notice if this device is being read even though the security administrator is not running the data reduction program for the audit log. It might also be possible to disable the read head entirely. Any type of write only device, such as a laser disk writer, would also provide the required assurance that the audit log has not been modified. If the device has no read hardware, then there is the additional assurance that it has not been read or altered.

The most common approach to protection of the audit log will use a combination of software and hardware assurance. In a typical application, either a double buffer or a circular buffer is used to accumulate audit log data before writing it to the logging device. The use of a double buffer is recommended since it is less likely than a circular buffer to lose data during heavy use. However, certain implementations of a double buffer may result in two objects to be protected, rather than one. The contents of the current buffer is the most vulnerable, and also would be the object of an intruder who has

just made an illegal access and is attempting to erase the evidence of the intrusion. The audit log buffer should be part of the TCB, protected by the most powerful hardware and software mechanisms available to the system designer. Further protection of the buffer might include keeping a running checksum so that if the buffer were modified, it would be unlikely that the intruder would be able to rewrite the checksum. When the buffer is written to the archival device, a bad checksum would cause a special audit record to be written, indicating that this buffer was suspect and recording the identity of all users on the system during the time the buffer was tampered with.

4. REDUCTION TOOLS

This section presents several alternative designs for reduction tools, which allow the system administrator to search the audit log for particular profiles of usage which might indicate an attempt to disclose protected information. Such tools are modeled on current technology query languages based on data base management systems, but have some features of a report generator system as well.

An audit reduction tool should interface to the security administrator as a combination of data base query language and report generator. Since a system with all audit flags turned on may generate a large amount of data, it is acceptable and usually necessary to encode the data to conserve space and minimize the processor time required to record the log records. Some provision must be made to expand the encoded fields to human readable form. This may be done as a preprocessor before the relevant log records are accessed by the data base system, or by a postprocessor which translates the encoded form only after a relevant record has been found. Such decoding is required both to present the data at the security administrator's terminal and to print out longer reports.

As an example of a typical query, the administrator might request the reduction tool to "Display all failed file access attempts made by user1, user2, and user3 between 0600 and 1200 today." This could be used to test the hypothesis that at least two of these individuals were passing data through a timing channel to each other. Clearly, such information could also be obtained by a series of queries, each producing a file of audit records. Then other types of software such as an editor could be used to extract common records. This type of system would satisfy the requirement within the *TCSEC*, but would be difficult to use. Given the availability of commercial data base management systems, easily learned and used reduction tools should be planned early in the development of a secure system. The system designers should consider whether new software should be written to perform the reduction function, or whether existing software should be modified or augmented to perform the task. It is possible that the reduction tools have access to files that should be protected at the highest level within the machine. If the audit log files are stored on the same hierarchical file system as other system records, and reports generated at the security administrator's request from those files might be stored at a lower level of protection, then the audit reduction tool must be considered a trusted process if this violates the security policy model of the system.

If the above considerations are examined, there are several choices that can affect the design of the reduction tools: dedicated storage device versus audit log stored within the system file structure; encoded versus full ASCII representation; processing within the same system versus offline processing; predefined report formats versus dynamic, security officer programmed report formats. Of course, a choice from one of these may preclude a selection in another area. For example, if the dedicated storage device has a low bandwidth, then the system must still be able to handle the audit log with all security relevant actions being recorded. If choosing to represent audit log entries as ASCII fields would exceed the bandwidth during normal operation, then encoded log entries must be chosen. System buffers might be able to handle the problem of temporary excess audit bandwidth requirements but if modeling shows the buffers

would grow unacceptably during average operations this solution should be unacceptable to the system designers. Other trade offs can be seen.

In the next several sections, several example systems are devised to illustrate the tradeoffs that must be considered in designing reduction tools.

4.1. Mainframe System

Consider a candidate B3 system with a large, powerful online storage system using disk storage. Assume the vendor company has its own data base management product. The equipment as installed is normally tended by a fulltime operator, and the system security officer at most sites will be an administrative staff person with little experience with computer programming.

An acceptable set of choices here would allow the audit log records to be stored within the filing system with the highest hierarchical security level, a special non-hierarchical category marked for "SSO Only." Because of the size and speed of the disk devices, full ASCII text could be stored into the audit records. Since the typical security officer will not be a programmer, the vendor decides that the proper interface is through the DBMS query language with predefined reports aggregated as described in section 3.2. Depending on the implementation of the DBMS, either the reports are entered into the audit log files in a format that is directly readable by the DBMS, or a special audit copy of the DBMS must be running as a process at all times so it can incorporate the records as they arrive and produce proper links on index fields.

Since fixed format reports are to be produced, the DBMS process can be set up with certain index fields that can be used to search the records with respect to a particular query. Since a suggested aggregation of records is by event type, the records should have an identifiable field called "event type" with a limited number of named types. This should be an indexed field so a search for all unsuccessful attempts to access an object, for example, can be easily found. As a secondary field, specification that only records generated since noon today could be used, although time is not an indexed field. A typical predefined format should be available for presenting information with "unsuccessful attempts" queries. A different form may be more appropriate with the "by user group" query. This would take full advantage of the power of the type of DBMS available on many mainframe computers.

The problem of what hierarchical level to give the report can be handled in several ways. If the report is only available at the security officer's own terminal, or at a printer running at maximum hierarchical level, then there is no problem since declassifying the report must be done manually by the security officer showing it to other personnel. If the summary reports are stored back in the filing system, either they must be stored at the same hierarchical level and non-hierarchical category as the audit logs, or the DBMS system must be considered a trusted process. The latter case is a problem for an A1 candidate system because this requires that the DBMS specification must be verified not to violate the system security model. Since most DBMS software is complex, and usually predates the operating system being

evaluated, this is not a viable solution. If there is a trusted editor available, it could be used to make a cleared copy of the report for dissemination to other users of the system.

4.2. Large Minicomputer

As a second example, consider a large minicomputer with limited disk capacity but one tape drive dedicated to audit records. The vendor has a DBMS product that runs on this system as an application. The typical security officer is on the system programming staff, and the system may be run without an operator. The disk file system is large and has a small granularity so that at peak operation with all audit checks enabled, the channel to the dedicated tape drive would be overwhelmed by full ASCII records.

A possible set of reduction tools here would require post processing of the taped audit log, either on the same system using a different or the same tape drive, or possibly on a different system running the same secure operating system product. Since the security officer is typically a system programmer, the system should allow both non-standardized queries and locally designed report formats. The post processing is required because the data are stored on the tape in encoded form, and since the security officer might choose any sort of query, or even want to inspect the audit log with an editor, the log should first be expanded into human readable form before it is made available to the user. This allows for a wide variety of processing; some installations may use a small set of predefined queries and report formats while other installations may have customized processing of the audit logs using locally written software. The post processing software should be supplied by the vendor although field names may be locally specifiable.

Since the audit log record is protected by being on a separate device, this system is suitable for any level C2 or higher system. The security officer could also be considered the originator of the summary reports and be allowed to determine their sensitivity level before storing them in the main file system i.e. disseminating them to other users.

4.3. Minicomputer with PC Offline

Some minicomputer systems have only limited disk storage and the addition of a large tape drive might make them economically unfeasible. One solution would be to connect the system through a dedicated channel to a personal microcomputer, which would store audit log records on a cartridge hard disk. Here the bandwidth to the microcomputer might be sufficient to take the maximum rate of log records, but since a small system such as this often runs unattended, encoded records should be stored in order to maximize the interval between disk cartridge changes. This type of system usually runs only a limited number of applications and the security officer will probably not be a programmer and may not want to learn even a DBMS query system language, so a special processor is required to run on the PC to access the audit log records. Numerous commercial products exist for database management on microcomputers, so the vendor should take advantage of these.

As an example, the PC could run a dedicated program that provides a menu driven interface to the security officer. With either the current hard disk cartridge in the PC connected to the minicomputer, or an archival hard disk cartridge running in an unconnected PC, the officer could select one or more criteria for menu search. The interface program could then translate this into the equivalent encoding used by the audit program, formulate a syntactically correct query to the DBMS, and interpet the DBMS output into human readable form before presenting it to the security officer. Since this is a fixed program, only certain report formats would be available. Since all processing is offline, dissemination of the reports is up to the security officer.

5. NEAR TERM TECHNOLOGY ADVANCES

Existing audit log mechanisms are essentially brute force efforts to inspect and check each auditable event. It is hoped that both hardware advances and improved software methodologies will make the audit mechanism easier for the system security administrator to use, easier for the system design team to develop, and more secure during operation of the secure systems it is designed to protect. In the hardware area, both write only archival devices and hardware mediated channel access are being developed. These are discussed below. In the software field, assurance increases with each increasing level of the criteria, so increased assurance in the audit mechanism seems tied to increasing requirements within the *TCSEC*. However, the lower level C2 and B1 systems would benefit most from increased assurance in the audit mechanism since this is the primary protection mechanism available to such systems. Anticipated software assurance requirements are discussed in Section 5.1.

5.1. Hardware Audit

There are two instances where newly designed hardware can benefit the audit logging mechanism. The first is write only devices, which then automatically prevent unauthorized read access to the audit log. It will not prevent appending data, such as logout mesages, to the audit log, but since a particular message cannot be located on the medium it is not possible to change a previously written message. By combining such devices with drivers that have very small buffers and by giving maximum protection to the buffer, maximum isolation of the audit log is assured. The log can subsequently be read by placing it on a read only device, which may be attached to a dedicated PC or other isolated computer, or be read on a read only device attached to the original system. This organization precludes immediate access to the audit data, which may be required by some applications. However, immediate access is not required by the *TCSEC*. Those events that require immediate access may be replaced by an alarm to the operator's console or security officer console.

Since most computer systems include a bus mechanism to transfer data between the central processing unit and the several memory units, various devices can be attached directly to this bus. The SCOMP system passes all memory requests through such a device, allowing hardware mediation of all data accesses. Future systems that follow this organization of physical components could include an audit mechanism program within the software or firmware of the mediation unit. Further isolation could be achieved by placing the archival device, possibly a removable disk pack, on a separate channel that communicated only with the mediation unit.

The widespread use of the personal computer (PC) suggests another possible hardware mechanism. The audit buffer could be sent over a medium speed communications line to a memory buffer in a PC, which could then stream the data to a hard disk. The administrator could then access the data using the personal computer with a program that interleaved buffering input data with searching the disk for audit log data. The log reduction tools could then be run only on the PC, effectively isolating them. Of course, these would be part of the TCB and subject to formal specification and verification at levels B3 and A1.

REFERENCES

[1] DoD Computer Security Center.
Trusted Computer System Evaluation Criteria.
Technical Report CSC-STD-001-83, United States Department of Defense, 1983.

[2] DoD Computer Security Center.
Guidance for Applying the DoD Trusted Computer System Evaluation Criteria in Specific Environments.
Technical Report CSC-STD-003-85, DoD Computer Security Center, 1985.

[3] S. L. Brand, J. D. Makey.
Department of Defense Password Management Guideline.
Technical Report CSC-STD-002-85, DoD Computer Security Center, 1984.

[4] DoD.
ADP Security Manual -- Techniques and Procedures for Implementing, Deactivating, Testing, and Evaluating Secure Resource-Sharing ADP Systems.
Technical Report 5200-28M, DoD, 1979.

[5] DoD.
Industrial Security Manual for Safeguarding Classified Information.
Technical Report 5220-22M, DoD, 1983.

[6] Bell, D. E., LaPadula, L. J.
Secure Computer Systems: Unified Exposition and Multics Interpretation.
Technical Report MTR-2997 Rev. 1, MITRE Corp., Bedford, MA, March, 1976.

[7] van den Berg, B., and H. Leenars.
Advanced Topics of a Computer Center Audit.
Computers and Security 3:171-185, 1984.

[8] DoD Computer Security Center.
Criterion Interpretation.
Technical Report C1-CI-01-84, DoD Computer Security Center, 1984.

[9] Gligor, V. D.
Guidelines for Trusted Facility Management and Audit.

[10] Landwehr, C. A.
Formal Models for Computer Security.
Computing Surveys 13(3):247-278, 1981.

[11] D. E. Denning.
A Lattice Model of Secure Information Flow.
Communications of the ACM 19(5):236-243, May, 1976.

[12] Bic, L.
A Protection Model and Its Implementation in a Dataflow System.
Communications of the ACM 25(9):650-658, 1982.

[13] D. I. Good, B. L. DiVito.
Using the Gypsy Methodology.
Technical Report, Institute for Computing Science, University of Texas, 1981.

[14] R. Kemmerer.
FDM - A Specification and Verification Methodology.
Technical Report SP-4088, System Development Corporation, 1980.

[15] D. H. Thompson.
AFFIRM Reference Manual.
Technical Report, Information Sciences Institute, University of Southern California, 1979.

[16] DoD Computer Security Center.
Criterion Interpretation.
Technical Report C1-CI-02-85, DoD Computer Security Center, 1985.

[17] DoD Computer Security Center.
Criterion Interpretation.
Technical Report C1-CI-07-84, DoD Computer Security Center, 1984.

APPENDIX A
AUDITABLE EVENTS

Since all systems are different, only some events will make sense for any particular system. This list identifies most of the events that are common to conventional modern-day computer systems, but is certainly not complete.

There are six general classes of auditable events, and any particular system may have others:

1. File system events
2. Other object events
3. User control events
4. Administrative and privileged events
5. Covert channel events
6. Invalid system states

1. File System Events

Creation of objects (C2)
Files, Directories, other file objects

Deletion of objects (specifically required at C2)
Files, Directories, other file objects

Modification of object access (C2)
Access bits, Access control lists

Modification of object security level (B1)
This is usually only possible for privileged users

Modification of object properties (B2)
Any properties other than access

Opening of objects (specifically required at C2)
File open, mapping into process address space, anything that causes an object to be manipulable by the process when it was not previously.

Closing of objects (B2)
All operations that are the reverse of "opening"

Unsuccessful access to objects (C2)
Any attempted operations that fail-- useful for detecting patterns in attempted access by users "hunting" for data to which they have unintentional access. This is required at B2 if the report of "object not found" events is identified as a covert channel.

2. Other Object Events

Operations on processes (C2)
Creation, deletion, etc.

Creation / Deletion of IPC channels (B2)
Operations that affect the ability to perform inter-process communication (IPC).

Unsuccessful IPC (C2)
Unsuccessful attempts to communicate between processes

Unsuccessful Machine Operations (C2)
Execution of privileged instructions or instructions that fail because of hardware-detected access violations. Failures resulting from invalid data formats, invalid register contents, etc., need not be audited.

Removable media requests (C2)
Requests by users for removable media, as distinct from the actual mounting and dismounting, which is performed by operators.

Removable media events (C2)
Physical mounting and dismounting of tapes and disks-- actions performed by operators in response to user requests

I/O device use (C2)
Operations performed by users to get, release, and control non-file system I/O devices (but not the I/O itself)

Communication channel use (C2)
Operations performed by users to get, release, and control communications channels, possibly excepting the one used by the user to communicate with the system, since that use is recorded elsewhere (and not the I/O itself)

Other object operations (as in section A.1)
Operations on other objects, depending on the system; for example modification of the access list for a tape volume owned by a user. For specific requirements, make the analogy to the file system operations listed in Section A.1.

3. User Control Events

Login and Logout (specifically required at C2)
All creation or destruction of user processes, or other requests made to that interface, such as reconnection to an existing user process.

Unsuccessful logins (C2)
Unsuccessful login or connection attempts. If a valid user ID is given, then this must be logged [8].

Identification and Authentication (specifically required at C2)
All operations not covered above that require the user to supply his password.

Operator login and logout (specifically required at C2)
Entry and exit of operators.

User and Group administration (specifically required at C2)
Creation and deletion of users and groups, and changes in group membership. These are usually administrative actions.

4. Administrative and Privileged Events

Operator activities (specifically required at C2)
All operator actions and all operator input. Operators will take some actions that can be detected by the hardware (such as assigning a tape drive to a user), but all operator typed input should be recorded as well.

Privileged operations (specifically required at C2)
All activities of the administrator.
All activities performed by a "superuser." If the administrator or operator can place the system (or his console) into some form of "privileged" mode, the facts of entry and exit must be recorded, and, if practical, the actions performed. Frequently, however, this is not practical, and in any case, the privileged user may have the capability to disable further recording. The object here is to record all operator, administrative, and privileged activities that occur during normal operation.

5. Covert Channel Events

Apparent covert channel use (specifically required at B2)
This is necessary to record events that might indicate unacceptably high rates of data transmission via the potential covert channel, not every instance of the event-- that is, an audit message need be generated only if the process performs the action often enough in a particular interval to transmit a significant amount of information. See the Covert Channel Guideline [1] for details.

Attempts to reference nonexistent objects (B2)
This is a common covert channel, and a special case of file system operations. The audit interpretation [17] holds that it is not necessary to audit such attempts for C2 or B1, even though they represent a "unsuccessful" attempt to access an object, since the object that was "unsuccessfully" accessed does not exist, and the only actual exposure caused by the attempt is a covert channel.

6. Invalid System States

Whenever the operating system detects an "impossible" state, this should be audited and appropriate action taken (repair the damage, crash the system, terminate the offending process, notify the security administrator).

APPENDIX B
EVENTS CAUSING REAL-TIME ALARMS

At the B3 and A1 levels, the system must be able to notify the security administrators of events that "may indicate an imminent violation of security policy" [TCSEC [1] Requirement 3.3.2.2.].

1. Required Alarms

The only classes of events that must be included are

1. Unsuccessful login attempts
2. Use of covert channels
3. Invalid system states identified during the evaluation

If a series of unsuccessful login attempts occurs, this usually indicates either a user who has forgotten his password, or a user who doesn't know the password-- that is, an intruder. It may also indicate a user attempting to log in from a location where he is not authorized to be, also a potential intrusion.

If the system detects an invalid state (such as a privileged program file in a place where privileged programs may not exist), this indicates either a system error or a successful penetration attempt. In either case, the security administrator must be notified. There is no requirement to detect all possible invalid system states, but the designers should ensure that checks are made at any identifiable potential weak spots.

2. Recommended Alarms

A series of unsuccessful file access attempts usually does not indicate an "imminent violation of security policy," for the simple reason that if the protection mechanism worked the first time, it will continue to work. Therefore, these events need not be included in the list that cause real-time alarms. However, although it may not indicate a policy violation, a series of unsuccessful access attempts may identify a user searching for files to which he does have (discretionary) access through unintentional action by the file's owner. Since this is a violation of intent (though not policy), it is useful if a real-time alarm system has a way to monitor unsuccessful access attempts.