NATIONAL COMPUTER

SECURITY CENTER

A GUIDE TO

UNDERSTANDING

CONFIGURATION MANAGEMENT

IN TRUSTED SYSTEMS

NCSC-TG-006-88
Library No. S-228,590

FOREWORD

This publication, "A Guide to Understanding Configuration Management in Trusted Systems", is being issued by the National Computer Security Center (NCSC) under the authority of and in accordance with Department of Defense (DoD) Directive 5215.1. The guidelines described in this document provide a set of good practices related to configuration management in Automated Data Processing (ADP) systems employed for processing classified and other sensitive information. Recommendations for revision to this guideline are encouraged and will be reviewed biannually by the National Computer Security Center through a formal review process. Address all proposals for revision through appropriate channels to:

National Computer Security Center
9800 Savage Road
Fort George G. Meade, MD 20755-6000

Attention: Chief, Computer Security Technical Guidelines

_____

Patrick R. Gallagher, Jr. 28 March 1988
Director
National Computer Security Center

ACKNOWLEDGEMENTS

Special recognition is extended to James N. Menendez, National Computer Security Center (NCSC), as project manager and primary author of this document.

Special acknowledgement is given to Grant Wagner, NCSC, and Dana Nell Stigdon, NCSC, for their constant help and guidance in the production of this document. Additionally, Dana Nell Stigdon, was responsible for writing the section on the Ratings Maintenance Program. Acknowledgement is also given to all those members of the computer security community who contributed their time and expertise by actively participating in the review of this document.

CONTENTS

FOREWORD i

ACKNOWLEDGEMENTS ii

CONTENTS iii

PREFACE v

1. PURPOSE 1

2. SCOPE 1

3. CONTROL OBJECTIVES 2

4. ORGANIZATION 3

5. OVERVIEW OF CONFIGURATION MANAGEMENT PRINCIPLES 4

    5.1 PURPOSE OF CONFIGURATION MANAGEMENT 4

6. MEETING THE CRITERIA REQUIREMENTS 5

    6.1 THE B2 CONFIGURATION MANAGEMENT REQUIREMENTS 5
    6.2 THE B3 CONFIGURATION MANAGEMENT REQUIREMENTS 6
    6.3 THE A1 CONFIGURATION MANAGEMENT REQUIREMENTS 6

7. FUNCTIONS OF CONFIGURATION MANAGEMENT 7

    7.1 CONFIGURATION IDENTIFICATION 7
        7.1.1 Configuration Items 8

    7.2 CONFIGURATION CONTROL 10
    7.3 CONFIGURATION STATUS ACCOUNTING 11
    7.4 CONFIGURATION AUDIT 12

8. THE CONFIGURATION MANAGEMENT PLAN 14

9. IMPLEMENTATION METHODS 16

    9.1 THE BASELINE CONCEPT 16
    9.2 CONFIGURATION MANAGEMENT AT MER, INC. 18
    9.3 THE CONFIGURATION CONTROL BOARD 20

10. OTHER TOPICS 23

   10.1 TRUSTED DISTRIBUTION 23
   10.2 FUNCTIONAL TESTING 24
   10.3 CONFIGURATION MANAGEMENT TRAINING 24
   10.4 CONFIGURATION MANAGEMENT SUPERVISION 25

11. RATINGS MAINTENANCE PROGRAM 26

12. CONFIGURATION MANAGEMENT SUMMARY 27

APPENDIX A: AUTOMATED TOOLS 29

A.1 UNIX (1) SCCS 29
A.2 VAX DEC/CMS 30

GLOSSARY 32

REFERENCES 34

PREFACE

Throughout this guideline there will be recommendations made that are not included in the Trusted Computer System Evaluation Criteria (TCSEC) as requirements. Any recommendations that are not in the TCSEC will be prefaced by the word "should," whereas all requirements will be prefaced by the word "shall." It should be noted that a TCSEC rating will only be based upon meeting the TCSEC requirements. Recommendations are made in order to provide additional ways of increasing assurance. It is hoped that this will help to avoid any confusion.

1. PURPOSE

The Trusted Computer System Evaluation Criteria (TCSEC) is the standard used for evaluating the effectiveness of security controls built into ADP systems. The TCSEC is divided into four divisions: D, C, B, and A, ordered in a hierarchical manner with the highest division, A, being reserved for systems providing the best available level of assurance. Within divisions C through A are a number of subdivisions known as classes, which are also ordered in a hierarchical manner to represent different levels of security in these classes.

For TCSEC classes B2 through A1, the TCSEC requires that all changes to the Trusted Computing Base (TCB) be controlled by configuration management. Configuration management of a trusted system consists of identifying, controlling, accounting for, and auditing all changes made to the TCB during its development, maintenance, and design. The primary purpose of this guideline is to provide guidance to developers of trusted systems on what configuration management is and how it may be implemented in the development and life-cycle of a trusted system. This guideline has also been designed to provide guidance to developers of all systems on the importance of configuration management and how it may be implemented.

Examples in this document are not to be construed as the only implementation that will satisfy the TCSEC requirement. The examples are merely suggestions of appropriate implementations. The recommendations in this document are also not to be construed as supplementary requirements to the TCSEC. The TCSEC is the only metric against which systems are to be evaluated.

This guideline is part of an on-going program to provide helpful guidance on TCSEC issues and the features they address.

2. SCOPE

An important security feature of TCSEC classes B2 through A1 is that there be configuration management procedures to manage changes to the Trusted Computing Base (TCB) and all of the documentation and tests affected by these changes. Additionally, it is recommended that such plans and procedures exist for systems not being considered for an evaluation or whose target evaluation class may be less than B2. The assurance provided by configuration management is beneficial to all systems. This guideline will discuss configuration management and its features as they apply to computer systems and products, with specific attention being given to those that are being built with the intention of meeting the requirements of the TCSEC, and to those systems planning to be re-evaluated under the Ratings Maintenance Program (RAMP) (see Section 11. RAMP).

Except in cases where there is a distinction between the

configuration management of a trusted system and an untrusted system, the word "system" shall be used as the object of configuration management, encompassing both the system and the TCB. It should be noted that the TCSEC only requires the TCB to be controlled by configuration management, although it is recommended that the entire system be maintained under configuration management.

3. CONTROL OBJECTIVES

The TCSEC gives the following as the Assurance Control Objective:

> "Systems that are used to process or handle classified or other sensitive information must be designed to guarantee correct and accurate interpretation of the security policy and must not distort the intent of that policy. Assurance must be provided that correct implementation and operation of the policy exists throughout the system's life-cycle."[1]

Configuration management maintains control of a system throughout its life-cycle, ensuring that the system in operation is the correct system, implementing the correct security policy. The Assurance Control Objective as it relates to configuration management leads to the following control objective that may be applied to configuration management:

> "Computer systems that process and store sensitive or classified information depend on the hardware and software to protect that information. It follows that the hardware and software themselves must be protected against unauthorized changes that could cause protection mechanisms to malfunction or be bypassed completely. [For this reason, changes to trusted computer systems, during their entire life-cycle, must be carefully considered and controlled to ensure that the integrity of the protection mechanism is maintained.] Only in this way can confidence be provided that the hardware and software interpretation of the security policy is maintained accurately and without distortion."[1]

4. ORGANIZATION

This document has been written to provide the reader with an understanding of what configuration management is and how it may be implemented in an ADP system.

For developers of trusted systems, this document also relates the TCSEC requirements to the configuration management practices that meet them. This document has been organized to illustrate the connection between practices and requirements through the use of a numbering convention for the TCSEC requirements. The

configuration management requirements have been broken down into 19 separate requirements in Section 6 of this document. The requirement number(s) will be located in parenthesis following its appropriate discussion, e.g., (Requirements 2, 15), signifies that the previous discussion dealt with TCSEC requirements 2 and 15 as stated in Section 6.

5. OVERVIEW OF CONFIGURATION MANAGEMENT PRINCIPLES

Configuration management consists of four separate tasks: identification, control, status accounting, and auditing. For every change that is made to an automated data processing (ADP) system, the design and requirements of the changed version of the system should be identified. The control task of configuration management is performed by subjecting every change to documentation, hardware, and software/firmware to review and approval by an authorized authority. Configuration status accounting is responsible for recording and reporting on the configuration of the product throughout the change. Finally, through the process of a configuration audit, the completed change can be verified to be functionally correct, and for trusted systems, consistent with the security policy of the system. Configuration management is a sound engineering practice that provides assurance that the system in operation is the system that is supposed to be in use. The assurance control objective as it relates to configuration management of trusted systems is to "guarantee that the trusted portion of the system works only as intended."[1]

Procedures should be established and documented by a configuration management plan to ensure that configuration management is performed in a specified manner. Any deviation from the configuration management plan could contribute to the failure of the configuration management of a system entirely, as well as the trust placed in a trusted system.

5.1 Purpose of Configuration Management

Configuration management exists because changes to an existing ADP system are inevitable. The purpose of configuration management is to ensure that these changes take place in an identifiable and controlled environment and that they do not adversely affect any properties of the system, or in the case of trusted systems, do not adversely affect the implementation of the security policy of the TCB. Configuration management provides assurance that additions, deletions, or changes made to the TCB do not compromise the trust of the originally evaluated system. It accomplishes this by providing procedures to ensure that the TCB and all documentation are updated properly.

6. MEETING THE CRITERIA REQUIREMENTS

This section lists the TCSEC requirements for configuration management. Each requirement for each class has been listed separately and numbered. Each number may be referenced to the requirement discussions that follow in this document. This section is designed to serve as a quick reference for TCSEC class requirements.

6.1 The B2 Configuration Management Requirements

Requirement 1 - "During development and maintenance of the TCB, a configuration management system shall be in place."[1]

Requirement 2 - The configuration management system shall maintain "control of changes to the descriptive top-level specification (DTLS)."[1]

Requirement 3 - The configuration management system shall maintain control of changes to "other design data."[1]

Requirement 4 - The configuration management system shall maintain control of changes to "implementation documentation"[1] (e.g., user's manuals, operating procedures).

Requirement 5 - The configuration management system shall maintain control of changes to the "source code."[1]

Requirement 6 - The configuration management system shall maintain control of changes to "the running version of the object code."[1]

Requirement 7 - The configuration management system shall maintain control of changes to "test fixtures."[1]

Requirement 8 - The configuration management system shall maintain control of changes to test "documentation."[1]

Requirement 9 - "The configuration management system shall assure a consistent mapping among all documentation and code associated with the current version of the TCB."[1]

Requirement 10 - The configuration management system shall provide tools "for generation of a new version of the TCB from the source code."[1]

Requirement 11 - The configuration management system shall provide "tools for comparisons of a newly generated TCB version with the previous version in order to ascertain that only the intended changes have been made in the code that will actually be used as the new version of the TCB."[1]

6.2 The B3 Configuration Management Requirements

The requirements for configuration management at TCSEC class B3 are the same as the requirements for TCSEC class B2. Although no additional requirements have been added, the configuration management system shall change to reflect changes in the design documentation requirements at class B3. This means that the additional documentation required for TCSEC class B3 shall also be maintained under configuration management.

6.3 The A1 Configuration Management Requirements

Requirements 2 through 11 are the same as those described in Section 6.1 for a class B2 rating. In addition the following requirements are added for class A1:

Requirement 12 - "During the entire life-cycle, i.e., during the design, development, and maintenance of the TCB, a configuration management system shall be in place for all security-relevant hardware, firmware, and software."[1]

Requirement 13 - The configuration management system shall maintain control of changes to the TCB hardware.

Requirement 14 - The configuration management system shall maintain control of changes to the TCB software.

Requirement 15 - The configuration management system shall maintain control of changes to the TCB firmware.

Requirement 16 - The configuration management system shall "maintain control of changes to the formal model."[1]

Requirement 17 - The configuration management system shall maintain control of changes to the "formal top-level specifications."[1]

Requirement 18 - The tools available for configuration management shall be "maintained under strict configuration control."[1]

Requirement 19 - "A combination of technical, physical, and procedural safeguards shall be used to protect from unauthorized modification or destruction the master copy or copies of all material used to generate the TCB."[1]

7. FUNCTIONS OF CONFIGURATION MANAGEMENT

7.1 Configuration Identification

Configuration management procedures should enable a person to "identify the configuration of a system at discrete points in time for the purpose of systematically controlling changes to the

configuration and maintaining the integrity and traceability of this configuration throughout the system life cycle."[4] The basic function of configuration identification is to identify the components of the design and implementation of a system. When it concerns trusted systems, this specifically means the design and implementation of the TCB. This task may be accomplished through the use of identifiers and baselines (see Section 9.1 The Baseline Concept). By establishing configuration items and baselines, the configuration of the system and its TCB can be accurately identified throughout the system life-cycle.

At TCSEC class B2, the TCSEC requires that "changes to the descriptive top-level specification, other design data, implementation documentation, source code, the running version of the object code, and test fixtures and documentation"[1] of the TCB be controlled by configuration management (Requirements 2, 3, 4, 5, 6, 7, 8). Configuration identification helps achieve this control. The TCSEC requires that each change to the TCB shall be individually identifiable so that a history of the TCB may be generated at any time. At TCSEC class A1, the requirements are extended to include that the "formal model...and formal top-level specifications" of the TCB shall also be maintained under the configuration management system (Requirements 16, 17).

The following is a sample list of what shall be identified and maintained under configuration management:

* the baseline TCB including hardware, software, and firmware

* any changes to the TCB hardware, software, and firmware since the previous baseline

* design and user documentation

* software tests including functional and system integrity tests

* tools used for generating current configuration items (required at TCSEC class A1 only)

Configuration management procedures should make it possible to accurately reproduce any past TCB configuration. In the event a security vulnerability is discovered in a version of the TCB other than the most current one, analysts will need to be able to reconstruct the past environment. This reconstruction will be possible to perform if proper configuration identification has been performed throughout the system life-cycle.

The TCSEC also requires at class B2 and above, that tools shall be provided "for generation of a new version of the TCB from the source code" and that there "shall be tools for comparing a newly generated version with the previous TCB version in order to ascertain that only the intended changes have been made in the code that will actually be used as the new version of the TCB"[1]

(Requirements 10, 11). These tools are responsible for providing assurance that no additional changes have been inserted into the TCB that were not intended by the system designer. Automated tools are available that make it possible to identify changes to a system online (see APPENDIX A: AUTOMATED TOOLS). Any changes, or suggested changes to a system should be entered into an online library. This data can later be used to compare any two versions of a system. Such online configuration libraries may even provide the capability for line-by-line comparison of software modules and documentation. At Class A1, the tools used to perform this function shall be "maintained under strict configuration control"[1] (Requirement 18). These tools shall not be changed without having to undergo a strict review process by an authorized authority.

7.1.1 Configuration Items

A configuration item is an uniquely identifiable subset of the system configuration that represents the smallest portion of the system to be subject to independent configuration management change control procedures. Configuration items need to be individually controlled because any change to a configuration item may have some effect upon the properties of the system or the security policy of the TCB.

Configuration items as they relate to the TCB, are subsets of the TCB's hardware, firmware, software, documentation, tests, and at class A1, development tools. Each module of TCB software for example, may constitute a separate configuration item. Configuration items should be assigned unique identifiers (e.g., serial numbers, names) to make them easier to identify throughout the system life-cycle. Proper identification plays a vital role in meeting the TCSEC requirement for class B2 that requires the configuration management system to "assure a consistent mapping among all documentation and code associated with the current version of the TCB"[1] (Requirement 9). Used in conjunction with a configuration audit, a consistent labeling system helps tie documentation to the code it describes. Not only does labeling each configuration item make them easier to identify, but it also increases the level of control that may be maintained over the entire system by making these items more traceable.

Configuration items may be given an identifier through a random distribution process, but, it is more useful for the configuration identifier to describe the item it identifies. Selecting different fields of the configuration identifier to represent characteristics of the configuration item is one method of accomplishing this. The United States Social Security number is a "configuration identifier" we all have that uses such a system. The different fields of the number identify where we applied for the Social Security card, hence describing a little bit about ourselves. As the configuration identifier relates to computer systems, one field should identify the system version

the item belongs to, the version of software that it is, or its interface with other configuration items. When using a numbering scheme like this, a change to a configuration item should result in the production of a new configuration identifier. This new identifier should be produced by an alteration or addition to the existing configuration identifier. A new version of a software program should not be identified by the same configuration item number as the original program. By treating the two versions as distinct configuration items, line-by-line comparisons are possible to perform.

Identifying configuration items is a task that should be performed early in the development of the system, and once something is designated as a configuration item, the design of that item should not change without the knowledge and permission of the party controlling the item. Early identification of configuration items increases the level of control that may be maintained over the item and allows the item to be traced back through all stages of the system development. In the event that a configuration item is not identified until late in the development process, accountability for that item in the early stages of the system development would be non-existent.

Configuration items may vary widely in complexity, size, and type, and it is important to choose configuration items with appropriate granularity. If the items are too large, the data identifying each one will overwhelm anyone trying to audit the system. If the items are too small, the amount of total identification data will overwhelm the system auditors.[2] The appropriate granularity for configuration items should be identified by each vendor and documented in the configuration management plan.

7.2 Configuration Control

"Configuration control involves the systematic evaluation, coordination, approval, or disapproval of proposed changes to the design and construction of a configuration item whose configuration has been formally approved."[5] Configuration control should begin in the earliest stages of the design and development of the system and extend over the full life of the configuration items included in the design and development stages. Early initiation of configuration control procedures provides increased accountability for the system by making its development more traceable. The traceability function of configuration control serves a dual purpose. It makes it possible to evaluate the impact of a change to the system and controls the change as it is being made. With configuration control in place, there is less chance of making undesirable changes to a system that may later adversely affect the security of the system.

Initial phases of configuration control are directed towards

control of the system configuration as defined primarily in design documents. For these, the Configuration Management plan shall specify procedures to ensure that all documentation is updated properly and presents an accurate description of the system and TCB configuration. Often a change to one area of a system may necessitate a change to another area. It is not acceptable to only write documentation for new code or newly modified code, but rather documentation for all parts of the TCB that were affected by the addition or change shall be updated accordingly. Although documentation may be available, unless it is kept under configuration management and updated properly it will be of little, if any use. In the event that the system is found to be deficient in documentation, efforts should be made to create new documentation for areas of the system where it is presently inadequate or non-existent.

To meet the TCSEC requirements though, configuration control shall cover a broader area than just documentation, and at Class B2 shall also maintain control of "design data, source code, the running version of the object code, and test fixtures"[1] of the TCB (Requirements 3, 5, 6, 7). A change to any of these shall be subject to review and approval by an authorized authority.

For TCB configuration items, those items shall not be able to change without the permission of the controlling party. At TCSEC class A1, this requirement is strengthened to require "procedural safeguards"[1] to protect against unauthorized modification of the materials used in the TCB (Requirement 19). These procedures should require that not only does the controlling party need to give permission to have a change performed, but that the controlling party performs the change on the master copy of the TCB that will be released. This ensures against changes being made to the master copy that are different than the approved changes.

The degree of configuration control that is exercised over the TCB will affect whether or not it meets the TCSEC requirements for configuration management. The configuration management requirements in the TCSEC require that a configuration management system be in place during the "development and maintenance of the TCB" at Class B2 (Requirement 1), and at Class A1, "during the entire life-cycle"[1] of the TCB (Requirement 12). A minimal configuration control system that would not be sufficient in meeting the TCSEC requirements, may only provide for review after a change has been made to the system. A system such as this may ensure that the change is complete and acceptable and may control the release of the change, but for the most part, the control exercised is little more than an after-the-fact quality assurance check. This system is certainly better than having no control system in place, but it would not meet the TCSEC requirements for configuration management. What is missing from this system that would bring it closer to the B2 requirements is control over the change as it is being made. The configuration control required by the TCSEC should provide for constant checking and approval of

a change from its inception, through implementation and testing, to release. The level of control exercised over the TCB may exceed that of the rest of the system, but it is recommended that all parts of the system be under configuration control.

In the case of a change to hardware or software/firmware that will be used at multiple sites, configuration control is also responsible for ensuring that each site receives the appropriate version of the system.

The point behind configuration control of the TCB is that all changes to the TCB shall be approved, monitored, and evaluated to provide assurance that the TCB functions properly and that all security policies are maintained.

7.3 Configuration Status Accounting

Configuration status accounting is charged with reporting on the progress of the development in very specific ways. It accomplishes this task through the processes of data recording, data storing, and data reporting. The main objective of configuration status accounting is to record and report all information that is of significance to the configuration management process. What is of significance should be outlined in the Configuration Management Plan. The establishment of a new baseline (see Section 9.1 THE BASELINE CONCEPT) or the meeting of a milestone is an example of what should be recorded as configuration status accounting information. The requirements in the configuration management plan should be viewed as the minimum and any events that seem relevant to configuration management should be captured and recorded in that they may prove to be useful in the future.

The configuration accounting system may consist of tracing through documentation manually to find the status of a change or it may consist of a database that can automatically track a change. As long as the information exists accurately in some form though, it will serve its purpose. The benefit of an online status accounting system is that the information may be kept in a more structured fashion, which would facilitate keeping it up to date. Being able to query a database for information concerning the status of a configuration change or configuration item would also be less cumbersome than sorting through notebook pages. Finally, the durability of a diskette or hard disk for storage outweighs that of a spiral notebook or folder, provided that it is properly backed up to avoid data loss in the event of a system failure.

Whichever system is used, it should be possible to quickly locate all authorized versions of a configuration item, add together all authorized changes with comments about the reason for the change, and arrive at either the current status of that configuration item, or some intermediate status of the requested item. The

status of all authorized changes being performed should be formulated into a System Status Report that will be presented at a Configuration Control Board meeting (see Section 9.3 THE CONFIGURATION CONTROL BOARD).

Configuration status accounting "establishes records and reports which enable proper logistics support, i.e., the supplying of spares, instruction manuals, training and maintenance facilities, etc. to be established."[5] The records and reports produced through configuration status accounting should include a current configuration list, an historical change list, the original designs, the status of change requests and their implementation, and should provide the ability to trace all changes.

7.4 Configuration Audit

Configuration auditing involves checking for top to bottom completeness of the configuration accounting information "to ascertain that only the [authorized] changes have been made in the code that will actually be used as the new version of the TCB."[1] (Requirement 11) When a change has been made to a system, it should be reviewed and audited for its effect on the rest of the system. This should include reviewing and testing all software to ensure that the change has been performed correctly.

Configuration auditing is concerned with examining the control process of the system and ensuring that it actually occurs the way it should. Configuration auditing for trusted systems verifies that after a change has been made to the TCB, the security features and assurances are maintained. Configuration audits should be performed periodically to verify the configuration status accounting information. The configuration audit minimizes the likelihood that unapproved changes have been inserted without going unnoticed and that the status accounting information adequately demonstrates that the configuration management assurance is valid.

"A complete audit should include tracing each requirement down through all functions that implement it to see if that requirement is met."[2] Furthermore, the configuration audit should also ensure that no additions were made that were not required. For the audit to provide a useful form of technical review, it should be predictable and as foolproof as possible, i.e., there should be specific desired results.

The configuration audit should verify that:

* the architectural design satisfies the requirements

* the detailed design satisfies the architectural design

* the code implements the detailed design

* the item/product performs per the requirements

* the configuration documentation and the item/product match

The main emphasis of configuration auditing is on providing the user with reasonable assurance that the version of a system in use is the same version that the user expects to be in use. Configuration audits ensure that the configuration control procedures of the configuration management system are being followed. The assurance feature of configuration auditing is provided through reasonable and consistent accountability procedures. All code audits should follow roughly the same procedures and perform the same set of checks for every change to the system.

8. THE CONFIGURATION MANAGEMENT PLAN

Effective configuration management should include a well-thought-out plan that should be prepared immediately after project initiation. This plan should describe, in simple, positive statements, what is to be done to implement configuration management in the system and TCB. A minimal configuration management plan may be limited to simply defining how configuration management will be implemented as it relates to the identification, control, accounting, and auditing tasks. The configuration management plan described in the following paragraphs is an example of a plan that goes into more detail and contains documentation on all aspects of configuration management, such as examples of documents to be used for configuration management, procedures for any automated tools available, or a Configuration Control Board roster (see Section 9.3 THE CONFIGURATION CONTROL BOARD). The configuration management plan should contain documentation that describes how the configuration management "tasks are to be carried out in sufficient detail that anyone involved with the project can consult them to determine how each specific development task relates to CM."[2]

One portion of the configuration management plan should define the roles played by designers, developers, management, the Configuration Control Board, and all of the personnel involved with any part of the life-cycle of the system. The responsibilities required by all those involved with the system should be established and documented in the configuration management plan to ensure that the human element functions properly during configuration management. A list of Configuration Control Board members, or the titles of the members should also be included in this section.

Any tools that will be available and used for configuration management should be documented in the configuration management plan. At TCSEC class A1, it is required that these tools shall

be "maintained under strict configuration control"[1] (Requirement 18). These tools may include forms used for change control, conventions for labeling configuration items, software libraries, as well as any automated tools that may be available to support the configuration management process. Samples of any documents to be used for reporting should also be contained in the configuration management plan with a description of each.

A section of the Configuration Management Plan should deal with procedures. Since the main thrust of configuration management consists of the following of procedures, there needs to be thorough documentation on what procedures one should follow during configuration management. The configuration management plan should provide the procedures to take to ensure that both user and design documentation are updated in synchrony with all changes to the system. It should include the guidelines for creating and maintaining functional tests and documentation throughout the life of the system. The configuration management plan should describe the procedures for how the design and implementation of changes are proposed, evaluated, coordinated, and approved or disapproved. The configuration management plan should also include the steps to take to ensure that only those approved changes are actually included and that the changes are included in all of the necessary areas.

Another portion of the configuration management plan should define any existing "emergency" procedures, e.g., procedures for performing a time sensitive change without going through a full review process, that may override the standard procedure. These procedures should define the steps for retroactively implementing configuration management after the emergency change has been completed.

The configuration management plan is a living document and should remain flexible during design and development phases. Although the configuration management plan is in place to impose control on a project, it should still be open to additions and changes as designers and developers see fit. This is not to say that the configuration management plan is only a guide and need not be followed, but that modifications should be able to occur. If the plan is not followed, there is no way it will be able to provide the appropriate assurances. In the event that a change is needed to the configuration management plan, the change should be carefully evaluated and approved. In changes to the configuration management plan of a trusted system this evaluation shall ensure that the security features and assurances supported by the plan are still maintained after the change has been implemented.

9. IMPLEMENTATION METHODS

This section discusses implementation methods for configuration

management that may be used to meet some of the requirements of the TCSEC. Section 9.1 discusses the baseline concept as a method of configuration identification. The baseline concept utilizes the features of configuration management spoken of previously, but divides the life-cycle of the system into different baselines.

Section 9.2 illustrates how a fictitious company, MER, Inc., conducts configuration management. They are attempting to meet the TCSEC requirements for a B2 system.

Section 9.3 discusses the concept of a Configuration Control Board (CCB) for carrying out configuration control. A CCB is a body of people responsible for configuration control. This concept is widely used by many computer vendors.

9.1 The Baseline Concept

Baselines are established at pre-selected design points in the system life-cycle. One baseline may be used to describe a specific version of a system, or in some configuration management systems a single baseline may be defined at each of several major milestones. Baselines should be established at the discretion of the Configuration Control Board and outlined in the configuration management plan. In cases where several baselines are established, each baseline serves as a cutoff point for one segment of development, while simultaneously acting as the step off point for another segment. The characteristics common to all baselines are that the design of the system will be approved at the point of their establishment and it is believed that any changes to this design will have some impact on the future development of the system.

Baseline management is one technique for performing configuration identification. It identifies the system and TCB design and development as a series of phases or baselines that are subject to configuration control. Used in conjunction with configuration items, this is another effective way to identify the system and its TCB configuration throughout its life-cycle.

"For each different type of baseline, the individual components to be controlled should be identified, and any changes that update the current configuration should be approved and documented. For each intermediate product in the development [life-cycle] there is only one baseline. The current configuration can be found by applying all approved changes to the baseline."[2]

In a system defining several baselines for different stages of development, these baselines or milestones should be established at the system inception to serve as guides throughout the development process. Although specific baselines are established in this case, alternatives may be recommended to

promote greater design flexibility or efficiency. The number of baselines that may be established for a system will vary depending upon the size and complexity of the system and the methods supported by the designers and developers. It is possible to establish multiple baselines existing at the same time so long as configuration management practices are applied properly to each baseline. The following example will discuss the baseline concept using three common baseline categories: functional, allocated, and product. It should be emphasized that these are simply basic milestones and baselines should be established depending upon the decisions of the designers and developers.

The first baseline, the functional baseline, is established at the system inception. It is derived from the performance and objectives criteria documentation that consists of specifications defining the system requirements. Once these specifications have been established, any changes to them should be approved.

The requirements produced in the functional baseline may be divided and subdivided into various configuration items. Once it has been decided what the configuration items will be, each of the items should be given a configuration identifier. From the analysis of the system requirements the allocated baseline will be established. This baseline identifies all of the required functions with a specific configuration item that is responsible for the function. In this baseline, an individual should be charged with the responsibility for each configuration item. All changes affecting specifications defining design requirements for the system or its configuration items as stated in the allocated baseline should require approval of the responsible individual.

The final baseline, the product baseline, should contain that version of the system that will be turned over for integration testing. This baseline signifies the end of the development phase and should contain a releasable version of the system.

The baseline example mention earlier in which one baseline is established for a single version of a system entails the same reasoning as the functional, allocated, and product baseline example. The system established as a baseline in the single baseline example will need to have an approved design before being placed under configuration control. Prior to the design approval, the system design will have to have undergone some type of functional review and a process that would allocate these functions to various configuration items. Although the early processes of the design will not be as formal in the single baseline example as they are when the early tasks are individually defined, the system will still benefit from being under the control of configuration management as a baseline. The main point of establishing any baseline is controlling changes to that baseline by requiring any changes to it to have to undergo an established change control process.

9.2 Configuration Management at MER, Inc.

MER, Inc., is a manufacturer of computer systems. Their latest project consists of building a system that will meet the B2 requirements of the TCSEC. In the past, their configuration management has only consisted of quality assurance checks, but to meet the B2 requirements they realize that they will need to have specific configuration management procedures in place during the development and maintenance of the system.

The project manager was assigned the task of writing the configuration management procedures and elected to present them in a configuration management plan. After doing some research on what should be contained in the configuration management plan, he proceeded to write a plan for MER, Inc. The configuration management plan that was written listed all of the steps to be followed when carrying out configuration management for the system. It described the procedures to be followed by the development team and described the automated tools that were going to be used at MER, Inc. for configuration management. These tools consisted of an online tracking data base to be used for status accounting, an online data base that contained a listing of all of the items under configuration control, and automated libraries used for storing software. Before development began, all of the development team was responsible for reading the configuration management plan to ensure that they were aware of the procedures to be followed for configuration management.

As the system was developed, the TCB hardware, software, and firmware were labeled using a configuration item numbering scheme that had been explained in the configuration management plan. In addition, the documentation and tests accompanying these items were also given configuration item numbers to assure a consistent mapping between TCB code and these items. All of the configuration item numbers and a description of the items were stored in a data base that could be queried at any time to derive the configuration of the entire system. Software and documentation were stored in a software library where they could be retrieved and worked on without affecting the master versions. The master copies of all software were stored in a master library that contained the releasable versions of the software. Both of these libraries are protected by a discretionary access control mechanism to prevent any unauthorized personnel from tampering with the software.

During the development of the system, changes were required. The procedures for performing a change under configuration control are described in the configuration management plan. These are the same procedures that will remain in effect throughout the life-cycle of the system. For each proposed change, a decision has to be made by management whether or not the change is

feasible and necessary. MER, Inc. has an online forum for reviewing suggested changes. This forum makes it possible for all of the members of the development team to comment on how the proposed change may affect their work. Management would often consult this forum to help arrive at their final decision.

After a decision was made, a programmer was assigned to perform the change. The programmer would retrieve the most recent version of the software from the software library and proceed to change it. As the change was being performed, the changes were entered into the online tracking data base. This made it possible for members of the development team to query this data base to find the current status of the change at any time. After the change had been performed it was tested and documented, and upon successful completion it was forwarded to a reviewer. This reviewer was the software manager, who was the only person authorized to approve a changed version for release. After the change was approved for release, the changed version was stored in the master library and a second copy was stored in the software library. Each change stored in these libraries was given a new configuration identification number. A tool was available at MER, Inc. that made it possible to identify changes made to software. It compared any two versions of the software and provided a line-by-line listing of the differences between the two.

It was realized at the beginning of the development process that there would be times when critical changes would need to be performed that would not be able to undergo this review process. For these changes, emergency procedures had been listed in the configuration management plan and a critical fix library was available to record critical changes that had occurred since a release.

A control process for changes to the TCB hardware was also provided for in the configuration management plan. The procedures ensured that changes to the TCB hardware were traceable and did not violate the security assumptions made by the TCB software. Similar to software changes, all hardware changes were reviewed by the project manager before being implemented.

After a change is made to the TCB software, MER, Inc. performs a configuration audit to verify the information that exists in the tracking data base. Whether or not a change is performed, the configuration management plan at MER, Inc. specifies that a configuration audit be performed at least once a month. This audit compares the current master version with the status accounting information to verify that no changes have been inserted that were not approved.

This configuration management plan encompasses the descriptive top-level specification (DTLS), implementation documentation, source code, object code, test fixtures, and test documentation,

and has been found to satisfy the TCSEC requirements for configuration management at class B2.

9.3 The Configuration Control Board (CCB)

Configuration control may be performed in different ways. One method of configuration control that is in use by systems already evaluated at TCSEC Class B2 and above is to have the control carried out by a body of qualified individuals known as the Configuration Control Board (CCB), also known as the Configuration Change Board. The Board is headed by a chairperson, who is responsible for scheduling meetings and for giving the final approval on any proposed changes. The membership of the CCB may vary in size and composition from organization to organization, but it should include members from any or all of the following areas of the system team:

* Program Management
* System Engineering
* Quality Assurance
* Technical Support
* Integration and Test
* System Installation
* Technical Documentation
* Hardware and Software/Firmware Acquisition
* Program Development
* Security Engineering
* User Groups

The members of the CCB should interact periodically, either through formal meetings, electronic forums, or any other available means, to discuss configuration management topics such as proposed changes, configuration status accounting reports, and other topics that may be of interest to the different areas of the system development. These interactions should be held at periodic intervals to keep the entire system team up-to-date with all advancements or alterations in the system. The Board serves to control changes to the system and ensures that only approved changes are implemented into the system. The CCB carries out this function by considering all proposals for modifications and new acquisitions and by making decisions regarding them.

An important part of having cross representation in the CCB from

various groups involved in the system development is to prevent "unnecessary and contradictory changes to the system while allowing changes that are responsive to new requirements, changed functional allocations, and failed tests."[2] All of the members of the Board should have a chance to voice their opinions on proposed changes. For example, if system engineering proposes a change that will affect security, both sides should be able to present their case at a CCB meeting. If diversity did not exist in the CCB, changes may be performed, and upon implementation may be found to be incompatible with the rest of the system.

The configuration control process begins with the documentation of a change request. This change request should include justification for the proposed change, all of the affected items and documents, and the proposed solution. The change request should be recorded, either manually or online in order to provide a way of tracking all proposed changes to the system and to ensure against duplicate change requests being processed.

When the change request is recorded, it should be distributed for analysis by the CCB who will review and approve or disapprove the change request. An analysis of the total impact of the change will decide whether or not the change should be performed. The CCB will approve or disapprove the change request depending upon whether or not the change is viewed as a necessary and feasible change that will further the design goals of the system. In situations where trusted systems are involved, the CCB shall also ensure that the change will not affect the security policy of the system.

Once a decision has been reached regarding any modifications, the CCB is responsible for prioritizing the approved modifications to ensure that those that are most important are developed first. When prioritizing changes, an effort should be made to have the changes performed in the most logical order whenever possible. The CCB is also responsible for assigning an authority to perform the change and for ensuring that the configuration documentation is updated properly. The person assigned to do the change should have the proper authorization to modify the system, and in trusted systems processing sensitive information, this authorization shall be required. During the development of any enhancements and new developments, the CCB continues to exert control over the system by determining the level of testing required for all developments.

Upon completion of the change, the CCB is responsible for verifying that the change has been properly incorporated and that only the approved change has been incorporated. Tests should be performed on the modified system or TCB to ensure that they function properly after the change is completed. The CCB should review the test results of any developments and should be the final voice on release decisions.

The use of a CCB is one way of performing configuration control,

but not every vendor may have the desire or resources to establish one. Whatever the preference, there should still be some way of performing the control processes described previously.

10. OTHER TOPICS

10.1 Trusted Distribution

Related to the configuration management requirements for trusted systems is the TCSEC requirement for trusted distribution at class A1 which states:

> "A trusted ADP system control and distribution facility shall be provided for maintaining the integrity of the mapping between the master data describing the current version of the TCB and the on-site master copy of the code for the current version. Procedures (e.g., site security acceptance testing) shall exist for assuring that the TCB software, firmware, and hardware updates distributed to a customer are exactly as specified by the master copies."[1]

Two questions that the trusted distribution process should answer are: (a) Did the product received come from the organization who was supposed to have sent it? and (b) Did the recipient receive exactly what the sender intended?

Configuration management assists trusted distribution by ensuring that no alterations are made to the TCB from the time of approved modification to the time of release. The additional configuration management requirement at A1 that supports this is, "A combination of technical, physical and procedural safeguards shall be used to protect from unauthorized modification or destruction the master copy or copies of all material used to generate the TCB"[1] (Requirement 19). This requirement calls for strict control over changes made to any versions of the TCB. The possibility that a change may not be performed as specified, or that a harmful modification may be inserted into the TCB should be considered and the authority to perform changes to the master copy should be restricted. A single master copy authority should be made responsible for ensuring that only approved and acceptable changes are implemented into the master copy.

Configuration status accounting records and auditing reports can provide accountability for all TCB versions in use. In the event of altered copies being distributed or "bogus" copies being distributed that were not manufactured by the vendor, configuration management records will be able to assess the validity and accuracy of all TCB versions. Trusted distribution displays the need for configuration control over all changes to the TCB. Without configuration control there would be no

accountability for the TCB versions distributed to the customer.

10.2 Functional Testing

"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."[1] The creation and maintenance of these functional tests is required to be part of the configuration management procedures. Test results and any affected test documentation shall be maintained under configuration management and updated wherever necessary (Requirements 7, 8). The tests should be repeatable, and include sufficient documentation so that any knowledgeable programmer will be able to figure out how to run them. The test plan for the system should be described in the functional specification (or other design documentation) for the TCB, along with descriptions of the test programs. The test plan and programs should be reviewed and audited along with the programs they test, although the coding standards need not be as strict as those of the tested programs.

It is not acceptable to only generate tests for code that was opened or replaced, but all of the portions of the TCB that were affected by the change should also be tested. The NCSC evaluators can provide a description of the security functional tests required to meet the TCSEC testing requirements, including the testing required as stated above for configuration management.

10.3 Configuration Management Training

Each new technical employee should receive training in the configuration management procedures that a particular installation follows. Experienced programmers, although they may be familiar with some form of configuration management, will also require training in any new procedures, i.e., an automated accounting system, that will be required to be followed. Training should be conducted either "by holding formal classes or by setting aside sufficient time for the reading of the company wide configuration standards."[2] New programmers should become familiar with the Configuration Management Plan before being allowed to incorporate any changes into the design baseline. It should be stressed that a failure to maintain the configuration management standards resulting from untrained employees, could prevent the system from receiving a rating.[2]

10.4 Configuration Management Supervision

A successful configuration management system requires the following of many procedures. Considering the demands made on

the system staff, errors may occur and shortcuts may be sought which will jeopardize the entire configuration management plan. A review process should be present to ensure that no single person can create a change to the system and implement it without being subject to some type of approval process. Supervisors, who are responsible for the personnel performing the change should be required to sign an official record that the change is the correct change.[2]

Proper supervision also provides assurance that whoever performs the change has the proper authorization to do so. Changes should not be performed by personnel that are not qualified to perform the change. Also, in systems that process sensitive information, the programmer performing the change shall possess the proper security clearance to perform the change.

Management itself must directly support the configuration management plan in order for it to work. It should not encourage cutting configuration management corners under any circumstances, e.g., due to scheduling or budgeting. Management should be willing to support the expenditure of money, people, and time to allow for proper configuration management.

11. RATINGS MAINTENANCE PROGRAM

The Ratings Maintenance Program (RAMP) has been developed by the NCSC in an effort to keep the Evaluated Products List (EPL) current. By training vendor personnel to recognize which changes may adversely affect the implemetation of the security policy of the system, and to track these changes to the evaluated product through the use of configuration management, RAMP will permit a vendor to maintain the rating of the evaluated product without having to re-evaluate the new version. Because changes from one version of an operating system to the next version may affect the security features and assurances of that operating system, configuration management is an integral part of RAMP. For a system to maintain its rating under this program, the NCSC shall be assured, through the vendor's configuration management procedures, that the changes made have not adversely affected the implementation of the security mechanisms and assurances of the system.

Each RAMP participant shall develop an NCSC approved Rating Maintenance Plan (RMPlan) which includes a detailed Configuration Management Plan (CMP) to support the rating maintenance process. This requirement applies to all systems participating in RAMP, regardless of class. For further information about the RAMP program and about configuration management requirements for RAMP, contact:

National Computer Security Center
9800 Savage Road

Fort George G. Meade, MD 20755©6000

Attention: Chief, Requirements and Resources Division

12. CONFIGURATION MANAGEMENT SUMMARY

The assurance provided by configuration management is beneficial to all systems. It is a requirement for trusted systems for classes B2 and above that a configuration management system "be in place that maintains control of changes to the descriptive top-level specification, other design data, implementation documentation, source code, the running version of the object code, and test fixtures and documentation"[1] (Requirements 1, 2, 3, 4, 5, 6, 7, 8). Although configuration management is a requirement for trusted systems for classes B2 and above, it should be in place in all systems regardless of class rating, or if the system has a rating at all.

Successful configuration management is built around four main objectives: control, identification, accounting, and auditing. Through the accomplishment of these objectives, configuration management is able to maintain control over the TCB and protect it against "unauthorized changes that could cause protection mechanisms to malfunction or be bypassed completely."[1] Even for those aspects of the system which are not security-relevant, configuration management is still a valuable method of ensuring that all of the properties of a system are maintained after a change. It is very important to the success of configuration management that a formal configuration management plan be adhered to during the life-cycle of the system.

A successful configuration management plan should begin with early and complete definition of configuration management goals, scope, and procedures. The success of configuration management is dependent upon accuracy. Changes should be identified and accounted for accurately, and after the change is completed, the change, and all affected parts of the system should be thoroughly documented and tested.

Configuration management provides control and traceability for all changes made to the system. Changes in progress are able to be monitored through configuration status accounting information in order to control the change and to evaluate its impact on other parts of the system.

An important part of having a successful configuration management plan is that the people involved with it must adhere to its procedures in order to keep all documentation current and the status of changes up-to-date.

With a firm and well documented configuration management plan in place, the occurrence of any unnecessary or duplicate changes

will be reduced greatly and any necessary changes that are required should be able to be identified with great ease. An effective configuration management system should be able to show what was supposed to have been built, what was built, and what is presently being built.

APPENDIX A: AUTOMATED TOOLS

Automated tools may be used to perform some of the configuration management functions that previously had to be performed manually. A data base management system, even with just a limited query system, may be used to perform the configuration audit and status accounting functions of configuration management. The principle behind using automated systems is that text, both from source code and other documents involved in the development of the system, can be entered into a Master Library and modified only through the use of the automated system. This prevents anyone from performing a change without having the proper authorization to access the configuration data base. "In general, only one program librarian, who should be the project manager or someone directly responsible to the manager, should have write access to the Master Library during development."[2]

A number of software developers have created software control facilities that are currently available to be used for configuration status accounting. A brief discussion of two of these systems follows.

A.1 UNIX (1) SCCS

"Under the Unix (1) system, the make utility, and the elements admin, get, prs, and delta, which comprise the Source Code Control System, provide a basic configuration accounting system. Initially a directory is created using the mkdir function. At this point, it is possible to use the owner, group, world protection scheme provided by Unix (1) to protect the directory. In addition a list of login identifiers is created which specifies who may update each element to be processed by SCCS." [2]

Following directory initiation, each document is entered using the admin -n function. Each entry that is made is referred to as an element. As each update is made to a new element, a new generation of that element, known as a delta, is created. The name of each element that is stored in a file by SCCS is preceded by "s.". If a file is added to the directory that does not contain this prefix, it is ignored by the SCCS function calls. When the admin function is called, a number of arguments may be specified that "specify parameters that may affect the file, and may be changed by a subsequent call to admin. The alogin argument is used to create the equivalent of an access control list by listing the login names of users who can apply the delta function to the element, thus creating either a new generation (delta) or variant branch."[2]

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

The initial release, or initial delta, of each code module is entered into the SCCS directory through the admin -n function, thus creating the Master Library. The programmer may update each module in the Master Library by using the get -e function "which indicates that the module will be edited and then the completed document will be reentered into the directory using the delta function. As long as the module being edited was extracted from the SCCS directory using get -e, it can be returned to the library using delta, and all necessary update information will be entered with it. The get function can be used to extract a copy of any document, but after it is edited it cannot be reentered into the library."[2]

"SCCS provides the capability to specify a software build by the way it assigns an SCCS Identification Number (SID) to each output of the delta function."[2] One can get any version of a text or source code by specifying the appropriate SID. "There are straightforward rules regarding how to specify the particular SID desired when get is called. If no SID is specified, the latest release and level is provided." The SID of the resulting call to delta is affected by the SID used when get -e is called.[2]

"The function prs allows for configuration accounting, since it extracts information from the s. files in the SCCS directory and prints them out for the user. Prs can be used to quickly create reports, listing one or two important values such as the last modified date for many SCCS files, or many values for one or two file. Larger reports can also be processed and created using an editor."[2]

A.2 VAX DEC/CMS

"VAX DEC/CMS [7] is also used to track a history of each text file stored in a CMS directory, but CMS does significantly more auditing and cross-checking than admin does. For example, if an editor is used directly to modify a file in a CMS directory, any further use by CMS of that file generates a warning meassage. Any files entered into a CMS directory by other than the CMS utility will cause CMS itself to issue a warning message when it is invoked for that directory. Otherwise, the process of configuration accounting is similar to SCCS.

The CMS CREATE LIBRARY function causes a directory to be set up, and initial logging to start. The project manager enters each element into the directory by using the CMS CREATE ELEMENT function. One must RESERVE an element of a library to modify it, and it can only be put back into the library using the REPLACE function. If someone else has RESERVEd an element between the original programmer's RESERVE and REPLACE calls, a warning is issued to both programmers and the occurrence is logged. To get a sample copy of the text, such as a program source, the FETCH function will generate the latest generation or any specified

generation of an element, but will not allow an edited copy to be reinserted into the library. The SHOW function can be used to audit the information about each element in the library.

Differences between SCCS and DEC/CMS appear concerning software builds. In Unix (1) a build must be either described in a makefile, or else each element to be used in a build must be retrieved from the SCCS directory using get, placed in another directory, and the makefile then may refer to these source files to create the executable build. In CMS, the process of selecting only a subset of source files, including some which are not the most current, is automated by the use of class and group mechanisms. To explain how this works, one must understand the CMS concepts of generations and variants. Each generation of a file corresponds to a Unix (1) delta. Generations are normally numbered in ascending order. CMS also has the capability of creating a variant development line to any generation by specifying in the REPLACE function a variant name. For example, if one RESERVEs generation 3 of an element, then performs a REPLACE/VARIANT = T, this will create generation 3T1 which may then be developed separately from generation 3. The first time this is used, the equivalent of an SCCS branch delta is created. Branches themselves can have branches, a capability that SCCS does not have.

A group can be defined within a CMS directory, using the CMS CREATE GROUP, and CMS INSERT ELEMENT functions. A group is composed of all generations, including variant generations, of all elements inserted into the group. Groups can be included within other groups. Groups can be defined with a non-empty intersection so that they have overlapping membership.

The CMS CREATE CLASS function, together with the CMS INSERT GENERATION function, can be used to specify the exact elements of a software build, and the DESCRIPTION file can then refer to the entire class by using the /GENERATION=classname qualifier on either the source or action line of a dependency rule. The makefile required by Unix (1) SCCS can be much more complex when it is required to describe a software build for intermediate testing."[2]

(1) Unix is a registered trade mark of Bell Laboratories

GLOSSARY

Automatic Data Processing (ADP) System - An assembly of computer hardware, firmware, and software configured for the purpose of classifying, sorting, calculating, computing, summarizing, transmitting and receiving, storing, and retrieving data with a minimum of human intervention.[1]

Baseline - A set of critical observations or data used for a comparison or a control. A baseline indicates a cutoff point in the design and development of a configuration item beyond which configuration does not evolve without undergoing strict configuration control policies and procedures.

Configuration Accounting - The recording and reporting of configuration item descriptions and all departures from the baseline during design and production.[2]

Configuration Audit - An independent review of computer software for the purpose of assessing compliance with established requirements, standards, and baselines.[2]

Configuration Control - The process of controlling modifications to the system's design, hardware, firmware, software, and documentation which provides sufficient assurance the system is protected against the introduction of improper modification prior to, during, and after system implementation.

Configuration Control Board (CCB) - An established committee that is the final authority on all proposed changes to the ADP system.

Configuration Identification - The identifying of the system configuration throughout the design, development, test, and production tasks.

Configuration Item - The smallest component of hardware, software, firmware, documentation, or any of its discrete portions, which is tracked by the configuration management system.

Configuration Management - The management of changes made to a system's hardware, software, firmware, documentation, tests, test fixtures, and test documentation throughout the development and operational life of the system.

Descriptive Top-Level Specification (DTLS) - A top-level specification that is written in a natural language (e.g., English), an informal program design notation, or a combination of the two.[1]

Firmware - Equipments or devices within which computer programming instructions necessary to the performance of the device's discrete functions are electrically embedded in such a

manner that they cannot be electrically altered during normal device operations.[3]

Formal Security Policy Model - An accurate and precise description, in a formal, mathematical language, of the security policy supported by the system.

Formal Top-Level Specification - A top-level specification that is written in a formal mathematical language to allow theorems showing the correspondence of the system specifications to its formal requirements to be hypothesized and formally proven.[1]

Granularity - The relative fineness or courseness by which a mechanism can be adjusted. The phrase "the granularity of a single user" means the access control mechanism can be adjusted to include or exclude any single user.[1]

Hardware - The electric, electronic, and mechanical equipment used for processing data.[3]

Informal Security Policy Model - An accurate and precise description, in a natural language (e.g., English), of the security policy supported by the system.

Software - Various programming aids that are frequently supplied by the manufacturers to facilitate the purchaser's efficient operation of the equipment. Such software items include various assemblers, generators, subroutine libraries, compilers, operating systems, and industry application programs.[6]

Tools - The means for achieving an end result. The tools referred to in this guideline are documentation, procedures, and the organizational body, i.e., the CCB, which all contribute to achieving the control objective of configuration management.

Trusted Computing Base (TCB) - The totality of protection mechanisms within a computer system -- including hardware, firmware, and software -- the combination of which is responsible for enforcing a security policy. A TCB consists of one or more components that together enforce a unified security policy over a product or system. The ability of a TCB to correctly enforce a security policy depends solely on the mechanisms within the TCB and on the correct input by system administrative personnel of parameters (e.g., a user's clearance) related to the security policy.[1]

REFERENCES

1. National Computer Security Center, DOD Trusted Computer System Evaluation Criteria, DOD, DOD 5200.28-STD, 1985.

2. Brown, R. Leonard, "Configuration Management for Development of a Secure Computer System", ATR-88(3777-12)-1, The Aerospace Corporation, 1987.

3. Subcommittee on Automated Information System Security, Working Group #3, "Dictionary of Computer Security Terminology", 23 November 1986.

4. Bersoff, Edward H., Henderson, Vilas D., Siegal, Stanley G., Software Configuration Management, Prentice Hall, Inc., 1980.

5. Samaras, Thomas T., Czerwinski, Frank L., Fundamentals of Configuration Management, Wiley-Interscience, 1971.

6. Sipple, Charles J., Computer Dictionary, Fourth Edition, Howard W. Sams & Co., 1985.

7. Digital Equipment Corporation, VAX DEC/CMS Reference Manual, AA-L372B-TE, Digital Equipment Corporation, 1984.