# FEDERAL CRITERIA

for

INFORMATION TECHNOLOGY SECURITY

VOLUME I

Protection Profile Development

Version 1.0

December 1992



NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
&
NATIONAL SECURITY AGENCY

# NOTES TO REVIEWERS

This is the first public draft of work in progress by the joint National Institute of Standards and Technology (NIST) and National Security Agency (NSA) Federal Criteria (FC) Project. This draft *Federal Criteria for Information Technology Security* is provided for preliminary review and comment by members of the national and international computer security community. The document will evolve into a new Federal Information Processing Standard (FIPS) intended principally for use by the United States Federal Government, and also by others as desired and appropriate. The FIPS is intended to replace the *Trusted Computer System Evaluation Criteria (TCSEC)* or "Orange Book."

Our objectives in presenting this draft material are threefold: first, to give the community a clear view of the FC Project's direction in moving beyond the TCSEC method of expressing requirements in order to meet new IT security challenges; second, to obtain feedback on the innovative approaches taken, the method of presentation, and granularity; and third, to make a substantial contribution to the dialogue among nations leading to the harmonization of IT security requirements and evaluations.

It is important to note a few things about this preliminary FC draft. First, it is a new and unpolished document and not intended for any purpose except review and comment. Organizations should not adopt any contents of this draft document for their use. It is anticipated that the document will undergo extensive revision as it works its way through the public FIPS approval process over the next year or two. Second, the FC is being distributed in two volumes. Volume I addresses the criteria development process and is intended principally for use by developers of protection profiles. The information in Volume I may also be of use to IT product manufacturers and product evaluators. Volume II presents completed IT product security criteria in the form of accepted protection profiles.

The protection profiles associated with the final FIPS will help consumers identify types of products that meet the protection requirements within their particular organizations and environments. However, the FIPS will be supplemented by a series of implementing guidance documents, many of which will be designed to help consumers make cost-effective decisions about obtaining and appropriately using security-capable IT products.

As a preliminary draft of the new FC-FIPS, this document is not intended for general distribution or compliance. The document should not be considered a complete or finished product. Your comments will be used by the Federal Criteria Working Group to help raise the maturity level of this material prior to being circulated for further public comment in the FIPS development process.

# ADDITIONAL NOTES TO REVIEWERS

Reviewers who provide substantive comments on the enclosed draft FC by March 31, 1993 will be invited to attend an Invitational Workshop on the Federal Criteria. This two-day workshop will be held in the last week of April 1993 in the Washington-Baltimore area at a location to be announced. All comments received by the cut-off date will be correlated into major themes for discussion by break-out groups at the workshop. The results will be used as input into the process of re-drafting the FC for a second round of comment prior to its being formalized as a FIPS.

Please send your comments (electronic format preferred) to **Nickilyn Lynch** at the U.S. National Institute of Standards and Technology (NIST), Computer Systems Laboratory (CSL).


**Phone:** **(301) 975-4267**
**FAX:** **(301) 926-2733.**

(Internet) Electronic Mail:

**lynch@csmes.ncsl.nist.gov**

Postal or Express Mail
(Hardcopy or 3.5”, 1.44M diskette in MSDOS, Macintosh, or Sun format):

**Federal Criteria Comments**
**Attn: Nickilyn Lynch**
**NIST/CSL, Bldg 224/A241**
**Gaithersburg, MD 20899**

Sincerely,

Eugene F. Troy
NIST Co-Manager
Federal Criteria Project

Ron S. Ross
NSA Co-Manager
Federal Criteria Project

## TABLE OF CONTENTS

## TABLE OF CONTENTS

TABLE OF CONTENTS

**TABLE OF CONTENTS**

## TABLE OF CONTENTS

**TABLE OF CONTENTS**

## LIST OF FIGURES

## LIST OF FIGURES

## LIST OF TABLES

# LIST OF TABLES

# Chapter 1.

# INTRODUCTION

## 1.1 Purpose

This Federal Information Processing Standard (FIPS) provides a basis for developing, analyzing, and registering criteria for information technology (IT) *product* security development and evaluation. It explains how to use provided generic requirements as building blocks to create unique sets of IT product security criteria called *protection profiles*.

This standard builds on national and international IT product security research and development by bringing together and extending many concepts of this previous work. The FIPS has four principal objectives.

a. **Develop an extensible and flexible framework for defining new requirements for IT product security.** IT product security criteria must respond to the challenges of extensible computing environments. The standard must provide a structured approach for specifying security requirements for IT products employed in such environments.

b. **Enhance existing IT product security development and evaluation criteria.** The fundamental principles of IT product security must be reviewed and renewed for application to new applications environments. The standard must address selected IT product security requirements of both Federal Government and private sector organizations.

c. **Facilitate international harmonization of IT product security development and evaluation criteria.** Producers of IT products competing in the international marketplace can benefit from a harmonized set of IT security development and evaluation criteria and an evaluation process that is economical, efficient, and predictable. The standard must meet U.S. Government and commercial security needs while recognizing that many of those needs are also shared by the government and commercial entities of other nations.

d. **Preserve the fundamental principles of IT product security.** The fundamental principles of IT product security developed during the past decades must be preserved. The standard must be compatible with previous IT product security requirements insofar as possible in order to protect previous investments in the technology.

## 1.2 Scope

This standard addresses the full spectrum of IT product security needs, to include *confidentiality*, *integrity*, and *availability*. Confidentiality requirements protect against inappropriate disclosure of information; integrity requirements ensure the correctness and appropriateness of information and/or its sources; and availability ensures that information is present and usable within reasonable time constraints.

This standard addresses the specification of *internal security controls* (protection mechanisms) that are implemented in the hardware, firmware, and software of an IT product. For these internal controls to be effective, however, adequate *external security controls* must be employed. IT product security is complemented by these external controls (which include physical, personnel, procedural, and administrative security measures) and by a separate certification and accreditation process. For an IT product, the external security measures constitute assumptions and boundary conditions that are part of the *environment* described in a protection profile. These environmental assumptions and boundary conditions are necessary to ensure IT products can be used in such a way as to meet identified security needs.

This standard distinguishes IT **product** requirements from IT **system** requirements. In general, an IT product is a hardware and/or software package that can be purchased as an off-the-shelf product and incorporated into a variety of systems. An IT system is generally constructed from a number of hardware and software components. For certain applications, it may be possible to purchase a single IT product that satisfies all customer requirements and, therefore, serve as a complete system. In most cases, however, at least some IT product customization and integration will be necessary to meet system specific requirements.

From a security perspective, the principal distinction between products and systems lies in what is certain about their operational environment. An IT product must be suitable for incorporation into many potential IT systems. Thus, the product developer can only make general assumptions about the operational environment of a system in which the product may be incorporated. These general assumptions include intended method of use and generalized threats within the environment. In contrast, an IT system must provide applications and meet the requirements of a specific group of end-users within a specific operational environment that has a specific set of threat scenarios.

This standard addresses IT product requirements only. The composition of multiple IT products into an IT system is beyond the scope of this standard. Guidance for profile and product composition will be addressed in future publications.

## 1.3 Audience

This document serves three primary customer groups with respect to IT product security:

a. **Consumers:** Individuals or groups responsible for specifying requirements for IT product security (e.g., policy makers and regulatory officials, system architects, integrators, acquisition managers, product purchasers, and end users).

b. **Producers:** Providers of IT product security (e.g., product vendors, product developers, security analysts, integrators, and value-added resellers).

c. **Evaluators:** Individuals or groups responsible for the independent assessment of IT product security (e.g., product evaluators, system security officers, system certifiers, and system accreditors).

Secondary audiences include technical educators, standards bodies, and the research and development community.

## 1.4 Organization of the Standard

The remainder of this FIPS is organized as follows. Chapter 2 describes the activities of IT security development. Chapter 3 addresses the form and content of protection profiles. Chapters 4, 5, and 6 provide detailed functional, development assurance, and evaluation assurance component requirements for use in constructing protection profiles. Chapter 7 is a guide to constructing protection profiles using the component requirements of Chapters 4 through 6. Several appendices provide additional supporting guidance.

This standard is part of a series of FIPS publications. Subsequent documents will be published as a Registry of Profiles representing profiles that have been developed, analyzed, and registered in accordance with this standard. Additional profiles will be added to the registry as consumer needs change and technology advances. Supporting guidelines for the standard will be published as part of this FIPS series or as other Federal agency publications.

# Chapter 2.

# IT SECURITY DEVELOPMENT

## 2.1 Overview

IT security development consists of three separate but related activities that begin with consumer specification of requirements for IT product security and end with installed IT systems incorporating products that have been approved to operate in a particular environment. The following list describes these activities, shown in Figure 1:

a. **Profile Development and Analysis.** IT product security requirements are specified in a structured format; analyzed for completeness, consistency, and technical correctness; and accepted into a registry of profiles.

b. **Product Development and Evaluation.** IT products are developed (or may already exist) in response to a profile and independently assessed to produce a rating regarding the product's conformance to a profile's specific security requirements.

c. **System Development and Certification.** One or more IT products are combined into an IT system that has been determined, from a security point of view, to be acceptable for use in a specific environment and accredited for operation.

This standard addresses the first of the three activities of IT security development, (i.e., profile development and analysis). Product development and evaluation as well as system development and certification are beyond the scope of this standard. Sections 2.4 and 2.5 briefly discuss these activities to establish their relationship to profile development.

In many cases, consumers will accept IT systems that contain unevaluated IT products, thus bypassing two of the activities of IT security development. This situation, however, places more demands on the system development process and the final certification and accreditation processes.

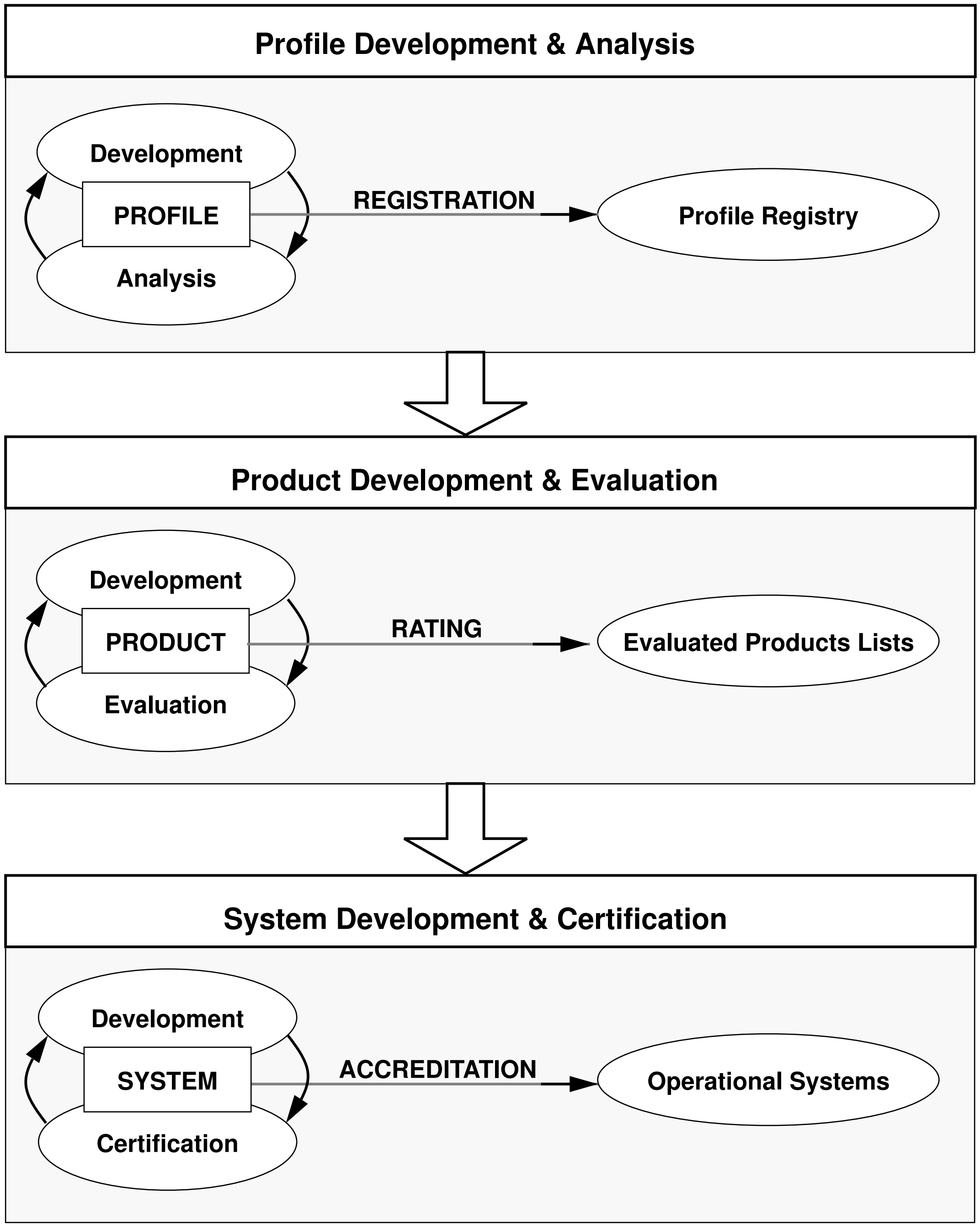



**Figure 1. IT Security Development Activities**

## 2.2 Functions and Assurance

This standard focuses on IT products that can potentially be used in many diverse environments. These products are required to support various organizational security policies and address a diverse set of security requirements by providing selected IT security features or services. Collectively, these security features or services are known as *protection functions.*

Specifying requirements for protection functions is a necessary but insufficient way to ensure consumer confidence that the resulting IT product will provide a viable solution to a *protection problem*. It is also necessary to consider the extent to which the protection functions can be relied upon. Are the functions appropriate to counter the threats? Are the functions sufficiently strong to counter the threats? Are the functions implemented soundly? Are there any threats not countered by the functions? The extent of this reliance is known as *assurance*. Assurance is the basis for consumer confidence or trust that an IT product is suitable, with respect to security, for its intended use.

Three sources of IT product assurance have been identified: protection functions built into the product, characteristics of how the product was designed and developed, and results of the independent examination of the product. These three aspects of IT product assurance (i.e., what it contains, how it was designed and developed, and how it was evaluated) are related. The evaluation activity examines the results of the IT product design, development, and implementation. Assurance requirements vary in accordance with organizational security policies, expected environment, and intended use of the IT product. Producers, consumers, and evaluators of IT product security perform different activities to obtain the requisite assurance.

## 2.3 Profile Development and Analysis

During profile development and analysis, consumers and/or producers define requirements for IT product security in a unifying structure called a *protection profile*. A protection profile contains IT product requirements for protection functions, development assurance, and evaluation assurance. These requirements can be framed in the context of a *rationale* statement, which provides the overall justification for the protection profile.

Acceptance of a newly developed protection profile requires that the profile be carefully scrutinized for its usefulness, both in content and form. It must also be analyzed for completeness, consistency, and technical correctness. Therefore, achieving profile acceptance will often require iteration as the initial profile is refined. Profile revision and analysis continues until an acceptable profile results. The profile can then be entered into a registry as basis for both product development and evaluation.

Some new profiles will have broad usefulness to the U.S. Government. These profiles are candidates to become Federal Information Processing Standards (FIPS). In these cases, the public FIPS development and approval process will encompass the profile analysis and registry mechanisms. For such profiles, NIST and NSA, with security professionals from the public and private sectors, will make use of invitational workshops and public review to provide the quality control and technical oversight that manages the proliferation of protection profiles. Chapter 3 provides additional detail that must be addressed during profile analysis, including profiles that are in the process of becoming FIPS. However, the specific details of that process are beyond the scope of this standard.

> *Editor's Note: Although the process of profile development and analysis is not fully mature, the final version of the Federal Criteria will successfully answer questions such as the following: Will all profiles be subjected to the same level of analysis? What methods of analysis and tools might be employed? Will profiles be subject to modification? How will new profiles be handled if they closely resemble existing profiles in the registry? Who will pay for profile analysis?*

## 2.4 Product Development and Evaluation

During product development and evaluation, a producer will incorporate protection functions into an IT product based on the requirements of a protection profile selected from the pool of registered profiles. Alternatively, a producer, who has identified a market for an IT product unrelated to one of the existing profiles, can undertake profile development and analysis.

The requirements in a protection profile should be product independent since many potential IT products may be able to satisfy the requirements of a particular profile. The comprehensive product description that explains how a specific IT product meets the requirements of a given protection profile is known as a *security target*. The security target is a specification and elaboration of the more general requirements in a protection profile and is, by definition, product dependent. The security target is the primary means of communicating specific product development information (evidence) to independent evaluators or to consumers. The development of product-specific security targets is beyond the scope of this standard.

Subsequent to development, an independent evaluation of an IT product may occur to produce a rating with respect to the product's conformance to the specific security requirements outlined in the protection profile. IT product evaluations may be accomplished by one of several evaluation authorities, just as profiles may be implemented by more than one producer. Consequently, specific details regarding evaluation processes are beyond the scope of this standard.

### 2.5 System Development and Certification

During system development and certification, IT products typically will be combined with other IT products into system configurations of varying degrees of complexity. The IT products used during system development may or may not have been formally evaluated. The completed IT systems will be subsequently employed in specific operational environments. An IT system must undergo an assessment and receive management approval prior to becoming operational. The assessment and management approval processes are known as system certification and accreditation, respectively.

IT system certification is conducted in support of the accreditation process. The extent to which a particular IT system meets a set of security requirements for its mission and operational environment is established by the comprehensive assessment of its internal and external security controls. In the Federal Government, a Designated Approving Authority (DAA) receives the resulting documentation to support the accreditation decision. In the private sector, this information might be provided to an equivalent designated management authority (e.g., corporate executive officer, department head, or division manager).

IT system accreditation is the official management decision to operate an IT system. The accreditation normally grants approval for the IT system to operate (1) in a particular security configuration, (2) with a prescribed set of countermeasures (administrative, physical, personnel, communications, emissions, and IT product internal security controls), (3) against a defined threat with stated vulnerabilities, (4) in a given operational context, (5) with stated interconnections to other systems, (6) at an acceptable level of risk for which the accrediting authority has formally assumed responsibility, and (7) for a specified period of time. The DAA formally accepts responsibility for the secure operation of the system and officially declares that a specified IT system will adequately protect against the identified threats through the continuous use of countermeasures. The accreditation decision affixes this responsibility with the DAA and shows that due care has been taken for security in accordance with applicable organizational security policies.

# Chapter 3.

# PROTECTION PROFILES

## 3.1 Overview

A protection profile is an abstract specification of the security aspects of a needed IT product. It is product independent, describing a range of products that could meet this same need. Required protection functions and assurances must be bound together in a protection profile, with a rationale describing the anticipated threats and intended method of use. The protection profile specifies requirements for the design, implementation, and use of IT products.

Protection profiles can be assembled from pre-specified or unique functional and assurance *components*. A functional component is a set of *rated* requirements for protection functions to be implemented in an IT product (see Chapter 4). An assurance component is a set of rated requirements for development and evaluation activities conducted by producers and evaluators during construction and independent assessment of an IT product (see Chapters 5 and 6). For convenience, groups of functional and assurance components can be assembled into predefined *packages* (see Appendixes A and B). During construction of the protection profile, additional dependencies must be considered between functions and assurances (see Chapter 7).

## 3.2 Sources of Protection Profiles

Consumers or producers within the Government or the private sector develop protection profiles in response to a specific need for information protection. Profile developers, or *sponsors*, with a unique security need could propose a protection profile for that need or, more typically, groups of sponsors having similar needs could combine to propose one profile that meets their common need. Multiple sponsors supporting a single profile is an effective way to demonstrate a larger market to potential IT product producers.

Unique protection profiles reflect the needs of diverse sets of sponsors. For example, a banker's association might propose a protection profile for secure electronic funds transfer, or the Department of Defense might propose a protection profile for military applications. A single protection may also apply to many IT products, showing the diversity of potential solutions for the requirements outlined in the profile.

A producer who identifies a market for IT product security can also propose a profile to give consumers a means of referring to a specific set of needs and to facilitate future evaluation against those needs. The protection profile is intended to respond to both the pull of consumer needs and to the push of advancing technology. Ultimately, the protection profile is a common reference among consumers, producers, and evaluators.

### 3.3 Protection Profile Contents

A protection profile contains five sections: descriptive elements, rationale, functional requirements, development assurance requirements, and evaluation assurance requirements. The **Descriptive Elements** section provides categorical and descriptive information necessary to identify, categorize, register, and cross-reference a protection profile in a registry of profiles. The narrative description is a brief characterization of the profile, including a description of the information protection problem to be solved. This section applies to all potential users of the profile to determine whether or not the profile is applicable to a consumer's information protection needs.

The **Rationale** section provides the fundamental justification for a protection profile, including threat, environment, and usage assumptions. It also presents a more detailed characterization of the protection problem to be solved by an IT product meeting the requirements of the profile. This section describes the protection problem in sufficient detail for producers to understand the range of potential solutions to the problem. It also provides information to consumers regarding how IT products that successfully solve this problem can be used to support an organization's security policy.

The **Functional Requirements** section establishes the information protection boundary that must be provided by an IT product. Expected threats to information within this boundary must be countered by functions inside the protection boundary. The more robust the expected threats, the greater the required strength of the protection functions. The IT product protection functions support an organization's security policy when coupled with certain assumptions about the product's intended use and anticipated operational environment.

The **Development Assurance Requirements** section covers all phases of an IT product's development, from the initial product design through implementation. Specifically, the development assurance requirements include the development process, development environment, and operational support requirements. In addition, since many assurance requirements are not readily testable, it is necessary to study IT product development evidence or documentation to verify that requirements have been met. Development evidence requirements are included in a protection profile to ensure that the producer generates and retains appropriate documentation during product development for subsequent analysis during evaluation and product maintenance.

The **Evaluation Assurance Requirements** section specifies the type and intensity of evaluation to be performed on an IT product developed in response to a particular protection profile. In general, for an IT product, the scope and intensity of evaluation vary with the expected threat, intended method of use, and assumed environment as defined by the profile developer in the rationale section. Table 1 summarizes the contents of a protection profile.

**Table 1. Protection Profile Structure**

| | |
|---|---|
| Descriptive Elements | Provides categorical and descriptive information necessary to uniquely identify, register, and cross-reference a protection profile in a registry of profiles. Includes a description of the information protection problem to be solved. |
| Rationale | Provides the fundamental justification for a protection profile, to include threat, environment, and usage assumptions. Addresses support for organization security policies. |
| Functional Requirements | Establishes the boundary of responsibility for information protection that must be provided by an IT product, such that expected threats to information within this boundary are countered. |
| Development Assurance Requirements | Specifies assurance requirements for all phases of an IT product's development from initial product design through implementation. Includes the development process, the development environment, operational support, and development evidence. |
| Evaluation Assurance Requirements | Specifies assurance requirements for the kind and intensity of evaluation to be performed on an IT product developed in response to a protection profile in accordance with the expected threat, intended method of use, and assumed environment. |

## 3.4 Protection Profile Development

The requirements for protection functions, development assurance, and evaluation assurance must be incorporated into a protection profile. These requirements, specified by the rated functional and assurance components, provide the basic building blocks for the definition of a protection profile. The components must be assembled into a consistent and coherent set that satisfies specific security goals of the anticipated environments of product use. The assembled components should counter expected threats, eliminate vulnerabilities, support security policies, and satisfy regulatory requirements defined in the anticipated environments of use.

Figure 2 shows that protection profile development consists of two stages: (1) an **environment security analysis** and (2) a **component requirement synthesis**. The environment security analysis addresses the identification of security requirements and provides information necessary for the development of the profile rationale. The component requirement synthesis addresses the selection of appropriate functional and assurance components for the profile. Developing a protection profile requires analysis of dependencies among the functional components, among assurance components, and between functional and assurance components (see Chapter 7).

### 3.4.1 Environment Security Analysis

During the environment security analysis stage, the sponsor (i.e., profile developer) derives a set of environment-specific security requirements based on expected threats and vulnerabilities; intended method of IT product use; environment assumptions; and policies, standards, regulations, or directives (if any). Although these requirements can be considered environment-specific, they derive from several potential environments of product use, and they capture the common security characteristics of those environments. The result of the security analysis, the environment-specific requirements, must characterize the environments of use in a demonstrable way.

The selection of environment-specific security requirements must be based on effectiveness of the security functions. The sponsor must show that the requirements in the protection profile satisfy the security objectives by countering the expected threats and eliminating the anticipated vulnerabilities. The effectiveness of the environment-specific requirements is a primary justification that must be provided in the profile rationale and an

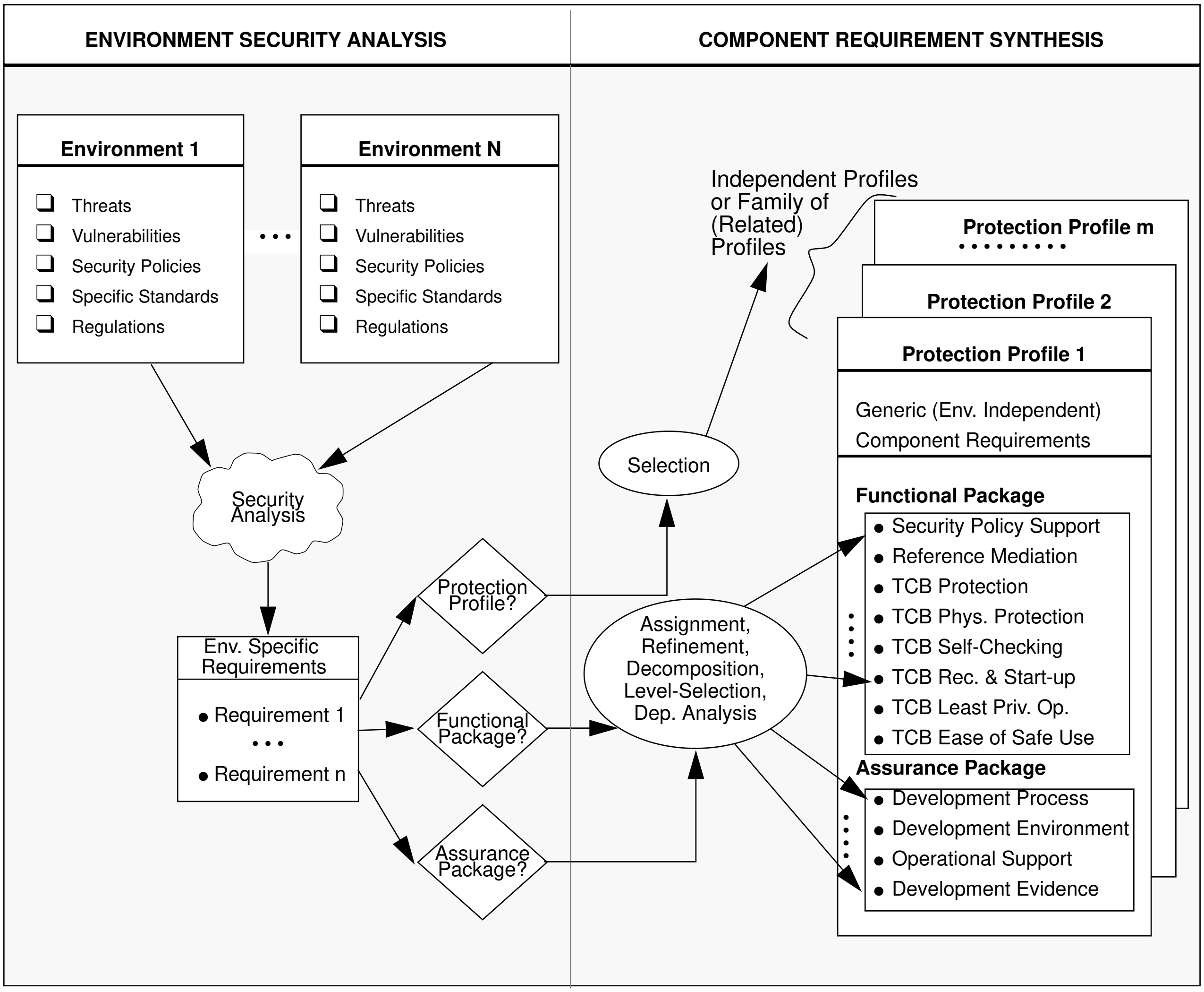



**Figure 2. Protection Profile Development**

important consideration in the acceptance of a profile. Other considerations, such as the utility and relevance of the anticipated environments of use, also apply to the profile analysis.

#### 3.4.1.1 Expected Threats

A *threat* is a classification of the capabilities, intentions, and attack methods of adversaries to exploit (or any circumstance or event with the potential to cause harm to) information or an information system. Harm to information or information systems due to threats may result because of absence or failure of functional controls. The consequences of threats may vary.

As suggested by Figure 3, the analysis of expected threats starts at the boundary of the IT

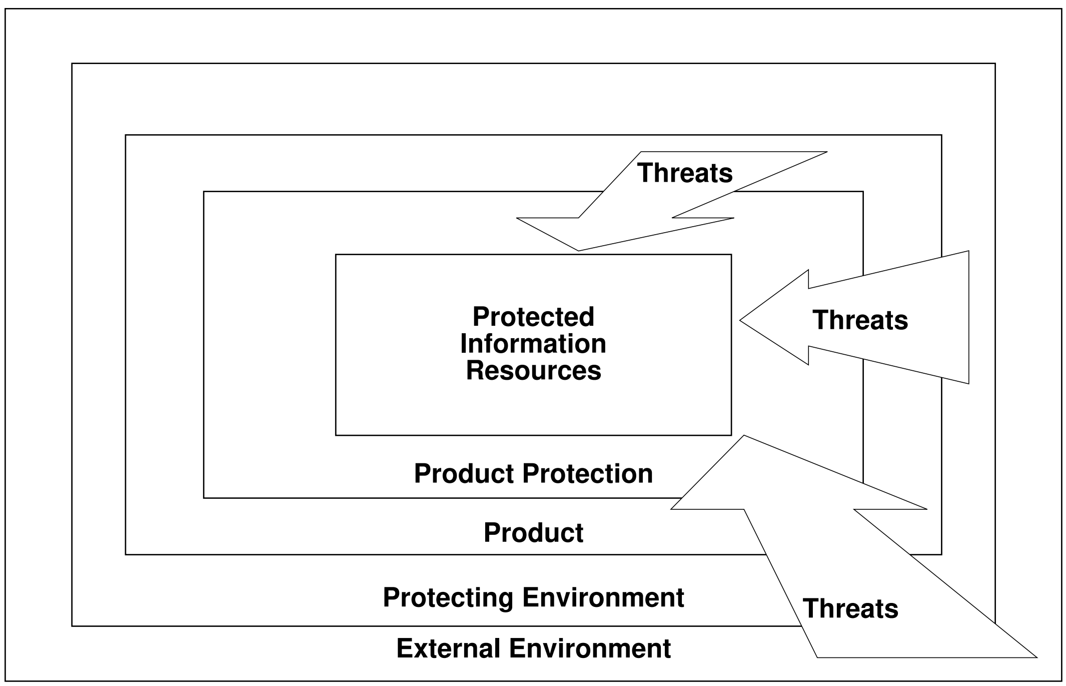



**Figure 3. Basis for Threat Analysis**

product's assumed environment. The scope of the analysis continues inward through the product's protection boundary to its protected information resources. One result of the analysis is the development of generic threat categories. These categories can be ordered according to *risk* (probability of occurrence) and level of severity. Appendix A provides a brief synopsis of common threats to information technology.

#### 3.4.1.2 Intended Method of Use and Environment

A protection profile must contain assumptions about the way the product will be used and the environment in which it will be placed. Assumptions should highlight significant constraints. For example, in some environments, routine product maintenance would be infeasible. These assumptions will enable the profile's users to understand the significance of the information being processed, the users and *administrators* involved in the information processing, the type of information processing, and the protection for the processing environment and its relationship to the users.

Sample rationales might include the following:

**Example 1:** This IT product generally will be used to process concurrent multiple levels of disclosure-sensitive and/or manipulation-sensitive information (i.e., national security information and/or information subject to organization internal controls and external regulation). In the assumed environment, sensitivity markings indicate the IT security controls that must be applied to protect the information. These sensitivity markings may be associated with objects that range in size from data elements to files.

**Example 2:** The users and administrators have access to multiple levels and types of information and processing resources. Access authorization is based on attributes, such as duties within roles, determination of need to know, trust indicators (such as individual clearances or job descriptions) and entry constraints (such as time, location, terminal, and port).

**Example 3:** Information is processed on centralized general-purpose shared computing resources allowing for both interactive and batch processing. The operational mode is concurrent multilevel processing. The user interface is generally expected to be window-based. Use of a database management system is anticipated. The database management system need not be a part of the product offering, but a description of how to integrate the database security interface must be provided.

**Example 4:** The processing resources of the IT product, including all terminations, will be located within user spaces that have *physical access controls*. A restricted access environment with unarmed guards should be assumed. The possibility of the environment becoming hostile should be considered (e.g., a U.S. Embassy in a foreign country). Networking may be anticipated, but is not required as part of the IT product offering.

### 3.4.2 Component Requirement Synthesis

During the second stage of protection profile development, the environment-specific security requirements must be used in conjunction with well-defined profile requirement construction rules to select and tailor the generic functional and assurance components provided in Chapters 4 through 6 of this standard. The resulting profile specifies the protection policy that must be supported within the IT product.

Not all environment-specific security requirements apply to the selection of the functional and assurance components. The environment-specific security requirements referred to in this section are those requirements that may be used to select functional and assurance components to be incorporated in a protection profile.

The selection of components also involves dependencies among components. Dependencies among functional components drive the selection of the functional components. Dependencies between functional and assurance components, and within the assurance components affect assurance component selection. Chapter 7 cites specific information on the techniques for constructing protection profiles and the dependency considerations between functional and assurance components.

## 3.5 Protection Profile Analysis

After a protection profile has been developed by a sponsor, it must be analyzed. Protection profile analysis ensures that the profile has the following characteristics: technical soundness, usefulness, evaluation capability, distinctness, and consistency.

The rationale section supports the profile analysis conducted to assess whether the vulnerabilities constituting a protection problem are adequately countered by the profile's proposed protection functions and assurances. The profile analysis determines that the risks identified in the protection problem have been reduced to an acceptable level. Profile analysis is important; many products may be created in response to a protection profile.

As described in the following sections, the goals of protection profile analysis are to ensure the following characteristics:

a. **Technical soundness**. The elements of a protection profile are technically sound and reasonably balanced, considering the profile rationale (threat, usage, and environment assumptions), and the functional and assurance requirements.

b. **Usefulness**. IT products built to meet the requirements of a protection profile will serve a useful purpose.

c. **Evaluation capability**. Implementation of a protection profile's requirements can be evaluated. Consumers, evaluators, and producers will understand how to determine that the profile requirements have been met by a specific IT product.

d. **Distinctness**. The protection profile is distinct, in that it does not duplicate a need adequately described by another profile.

e. **Consistency**. The protection profile is consistent with other profiles in form and level of detail.

### 3.5.1 Technical Soundness

Determining technical soundness is a crucial part of the analysis of a protection profile. The following three major characteristics should be considered:

a. **Strength or appropriateness of protection functions and assurances**. This characteristic is a judgment made by comparing the profile rationale (threat, usage, environment assumptions, and security policy) with the required protection functions and assurances. It must be determined that if the IT product is used as recommended, each stated threat will be successfully addressed through the prescribed combination of functional and assurance requirements, taking into account assumptions about the environment.

b. **Internal consistency of profile requirements**. This characteristic is a judgment based on an overall analysis of the protection profile to determine that the degree of required protection functions is commensurate with the degree of required assurance.

c. **Requirement dependencies**. This characteristic involves dependencies that may exist among requirements and whether any dependencies have not been considered in the development of the protection profile. These dependencies can be functional-functional, assurance-assurance, or functional-assurance. Dependency analysis must be complete and consistent.

Questions to be answered during the technical soundness analysis might include the following: Are the functional requirements sufficient to counter the expected threats? Are any threats not countered adequately addressed in environmental and/or other assumptions outside the domain of the IT product? Is the degree of assurance compatible with the threat expected and with the level of protection functions required? Have all stated dependencies been addressed? Have any dependencies been omitted? Are there inconsistencies in protection functions resulting from an examination of dependencies?

### 3.5.2 Usefulness

A management decision is necessary to commit the resources for conducting a profile analysis. Consumers and producers may determine the usefulness of a profile based on different criteria. Determining the profitability of a market is clearly a producer's responsibility. The protection profile analysis is not intended to interfere with the producer's business decisions. Usefulness analysis relies primarily on the rationale section of the profile to express the need with respect to threat, usage and environment assumptions.

The analysis also includes an assessment of development feasibility. Protection profiles requiring research and development efforts should be identified so that the profiles will not raise expectations for near-term implementations.

Questions to be answered during the usefulness analysis might include the following: Does this protection profile address a real problem? Does this protection profile differ significantly from existing profiles, or could the needs described in this profile be met adequately by another existing profile? If the demand is too small to support commercial off-the-shelf products, what factors could induce producers to develop products for a niche market? Approximately how large is the demand for these products? Is the protection profile readily implementable? Is the state of technology sufficiently developed or is basic or applied research and development necessary?

### 3.5.3 Evaluation Capability

The protection profile requirements must be reviewed to ensure that IT products intended to satisfy the requirements in the protection profile are capable of being evaluated. A protection profile may be used as the basis for product development. The evaluation capability analysis of the profile can pay off by clearly defining what is expected during the evaluation process. As far as possible, requirements in the protection profile should be stated in objective terms so the producer and the evaluator will be more likely to agree on their interpretation of the requirements. If certain requirements must be stated in subjective terms, clear and explicit guidelines should be presented to explain what factors should be considered to determine whether the requirements have been met. Requirements should be simple, declarative statements and for ease of reference, requirements should be numbered or otherwise indexed. These features will help to ensure that requirements will not be overlooked, either during product design and development or during evaluation.

Questions to be answered during the evaluation capability analysis might include the following: Is the purpose or objective of each requirement clear? What does each requirement contribute to the overall IT product security? Is the phrasing of each requirement clear and concise? Are the criteria for success for each requirement self-evident or explicitly provided? Is there a means by which it can be shown that each protection requirement has been established in the producer's IT product? Is each requirement objective? If a requirement is subjective, is it accompanied by objective factors to be considered when determining if the requirement has been met?

### 3.5.4 Distinctness

A new protection profile is compared with existing profiles to determine that it meets a unique need and that the requirements of no other existing profile address that same need. A protection profile, once registered, is not intended to be changed (except for editorial changes that would not affect any producers currently developing products to meet that profile specification). This constraint is intended to preserve a relatively stable and manageable set of requirements. New needs must be met by new profiles. Similar protection profiles are cross-referenced.

Questions to be answered during distinctness analysis might include the following: Do the presumed threats described in this profile very closely resemble those of any other existing profile? If yes, is there a significant difference between the two profiles? Do the required protection functions very closely resemble those of any other existing profile? If yes, does a significant difference exist between the two profiles? Do the functional requirements and rationale sections of the protection profile resemble another protection profile with different assurance requirements? If yes, can assurance requirements of the other profiles be changed?

### 3.5.5 Consistency

Protection profiles must be consistent with one another in form and level of detail. This review analyzes the profile to ensure that the properties associated with accepted protection profiles are present. A clear and convincing case must be made for allowing protection profiles whose form must differ from the norm.

Questions to be answered during consistency analysis might include the following: Is the protection profile complete? Are the objectives and expected threats discussed reasonably complete for the expected environments and intended method of use? Can the threats be shown to be mitigated by attributes of the IT product or its environment? Is the protection profile internally consistent in its level of protection? Are the form and degree of detail of the protection profile consistent with the form and degree of detail of other profiles?

## 3.6 Protection Profile Registration

To provide a relatively stable environment, a profile is intended never to be changed once it is registered. However, evaluation experience may identify errors and ambiguities that need to be corrected. New information protection requirements will be addressed by the development of new profiles.

Protection profiles that have completed analysis will be registered. Producers can select profiles for IT product implementation. The profiles in the public registry can also serve as templates for the development of new profiles.

> *Editor's Note: The specific details of profile registration are currently under development and have not been completed. It is envisioned that the profile registry could contain three independent registration types: (1) the complete protection profiles, (2) functional packages, and (3) assurance packages. Packages would identify any associated dependencies. The registration bodies that approve the inclusion of new protection profiles would also approve the registry of new packages for protection functions and assurance.*

# Chapter 4.

# FUNCTIONAL REQUIREMENTS

## 4.1 Overview

The functional requirements presented in this chapter enable the definition of different protection profiles that can be used in different environments of IT product use and address different threats. These requirements allow protection profile extension and refinement, which may become necessary as technology changes and as experience is gained. They also enable the harmonization of this standard with existing standards, such as the *Canadian Trusted Computer Product Evaluation Criteria (CTCPEC),* the *Information Technology Security Evaluation Criteria (ITSEC),* and the *Trusted Computer System Evaluation Criteria (TCSEC)*. Thus, these requirements allow the definition of protection profiles that closely capture the functional characteristics of IT products evaluated under the existing standards.

The functional requirements defined in this chapter are grouped into components of *trusted computing bases* (TCBs). The TCB is the totality of an IT product's elements, comprising the hardware, firmware, and software code and data structures responsible for enforcing the product's protection functions. Thus, a functional component is a set of requirements levied on either (1) one or more TCB functions that can be invoked through the TCB interface (e.g., system call, supervisor call) or (2) one or more internal modules or sections of code and data structures of a TCB function.

The functional components defined in this standard are based on the premise that the TCB is the only part of the product that needs to be analyzed and evaluated to determine the protection characteristics of a product. For this reason, this standard need not define more than the desirable sets of functional components for TCBs. Since different functions of a TCB help counter different threats, the analysis of the TCB protection must identify the set of components that collectively helps counter a specified set of threats. To make a protection profile generally applicable to a wide set of products, the desirable components included in a protection profile must be specified in a product-independent manner, in terms of generic protection requirements, rather than by specific mechanisms that may vary from product to product. The threats countered by the TCB functional components also must be defined in generic terms rather than by specific threat instances that may vary from environment to environment.

The functional components presented in this standard are derived from existing security criteria and requirements of commercial and non-commercial environments. To address a wide variety of protection needs, each functional component is rated based on a set of well-defined parameters. This rating is intended to capture the desirable variations in the protection merits of component requirements. This rating can also help clarify the relationships between these requirements and those of existing standards.

This chapter is divided into four sections. The remainder of this section groups the functional components of a TCB into eight classes and describes the types of components in each class. The second section presents a description of each type of functional component in terms of the generic threats and vulnerabilities these components are intended to counter or eliminate. The third section presents the rated functional components. The last section includes a bibliography of useful literature references. (Appendix B presents the reference monitor concept and its role in product security, and Appendix C presents the components required in defining an access control policy.)

**Classes of TCB Functions:** Eight classes of TCB functions and associated components with distinct protection requirements are identified in the Taxonomy of TCB Functions (Figure 4). These classes are: (1) security policy support, (2) reference mediation, (3) TCB logical protection, (4) TCB physical protection, (5) TCB self-checking, (6) TCB start-up and recovery, (7) TCB privileged operation, and (8) TCB ease-of-use. It is important to note that all but the first class of the components listed above are sometimes considered to be operational assurances. However, a different point of view is taken in this standard for two reasons. First, if a protection relevant component requires that specific software, hardware, or firmware elements (i.e., code, data structures) be part of the TCB, then that component implements a necessary protection function, even if it only indirectly contributes to the overall protection of the product. Second, functional components actively help counter TCB external threats or eliminate vulnerabilities, whereas assurance components do not. Instead, the assurance components help identify and eliminate potential vulnerabilities caused by errors of omission, or commission, in TCB development, life cycle maintenance, and operation. Therefore, the items in component classes two through eight are categorized as functional components instead of operational assurances.

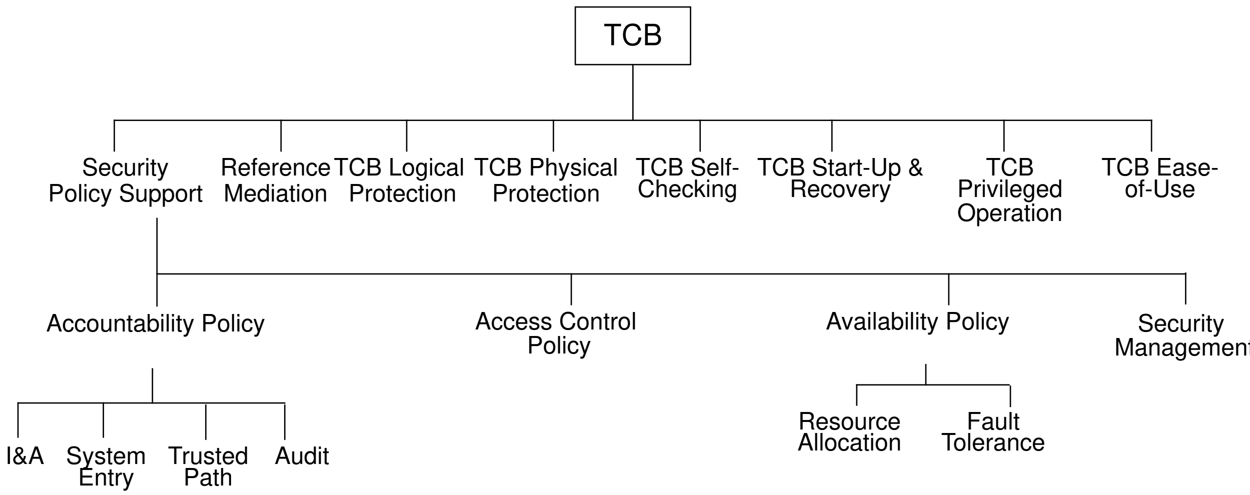



**Figure 4. Taxonomy of TCB Functions.**

**Security Policy Support**. This class of components defines four functional components for basic security policy support at the interfaces of typical TCBs: (1) accountability policy components, which include the functional components of identification and authentication (I&A), system entry control, trusted path, and audit, (2) an access control policy component, (3) availability policy components, which include resource allocation and fault tolerance components, and (4) security management components. The degree to which a product's TCB must implement the requirements of these functional components depends upon the threat environment assumed and the product's security objectives. Furthermore, the ability of a product's TCB to correctly support a set of organizational security policies depends jointly on (1) the product policies implemented by the TCB functions (e.g., these policies must be consistent with those of the organization), (2) the correct operation and input by system administrative personnel (e.g., system start-up or recovery must be performed properly; the user registration and the system entry parameters must be set properly), and (3) the actions of the unprivileged users themselves (e.g., choice of passwords, setting of an object's default access rights, distribution of access rights).

**Reference Mediation.** The requirements of this component ensure that all references issued by subjects external to the TCB (i.e., unprivileged subjects) to objects, resources, and services of a product are validated by the TCB in accordance with the security policies of the product. Satisfying the requirements of this component establishes *complete* reference mediation (i.e., a reference of a subject external to the TCB cannot circumvent the security policies of the TCB).

**TCB Logical Protection.** The requirements of this component ensure that at least one domain is available for the TCB's own execution, and that the TCB is protected from external interference and tampering (e.g., by modification of TCB code or data structures) by unprivileged subjects. Satisfying the requirements of this component makes the TCB self-protecting. Therefore, an unprivileged subject cannot modify or damage the TCB.

Note that the reference mediation and the TCB logical protection components include the first two requirements of the *reference validation mechanism* (see Appendix B). These two components, as well as the security policy support, are necessary for all protection profiles. The strong dependency of these two components on the development assurance components is defined by the third requirement of the reference validation mechanism (see Chapter 7; Appendix B).

**TCB Physical Protection.** The requirements of this component ensure that the TCB is either protected from physical tampering and interference or operates in a protected environment. Satisfying the requirements of this component causes the TCB to be packaged and used in such a manner that (1) physical tampering is detectable, or (2) resistance to physical tampering is measurable based on defined work factors. Without this component, the protection functions of a TCB lose their effectiveness in environments where physical damage cannot be prevented.

**TCB Self-Checking.** The requirements of this component ensure that hardware, firmware, or software are available to validate the correct operation of the TCB and the consistency of the TCB's protection data structures. Satisfying the requirements of this component allows the TCB to (1) detect corruption of protection-relevant code and data structures resulting from various mechanism failures and (2) initiate corrective action. This component is important because, unlike TCB protection, corruption of protection-relevant code and data structures resulting from mechanism failures can only be detected, not prevented.

**TCB Start-Up and Recovery.** The requirements of this component ensure that the TCB can determine that the IT product is started without protection compromise and can recover without protection compromise after a detected failure or other discontinuity. Satisfying the requirements of this component establishes that the initial and recovered states of a TCB satisfy the security policy, reference mediation, and TCB protection requirements. This component is important because the start-up TCB state determines the protection of subsequent states, and once the corruption of a protection-relevant data structure by a failure is detected, TCB recovery action becomes necessary.

**TCB Privileged Operation.** The requirements of this component ensure that TCB functions operate with the fewest privileges necessary to accomplish their purpose. Satisfying the requirements of this component causes identification of system privileges required by each TCB function and the addition of mechanisms that associate these privileges with specific TCB functions, modules, or actions. This component is important because it helps restrict the propagation of errors and failures.

**TCB Ease-of-Use.** The requirements of this component enable the use of the TCB by users, administrators, and their applications. Satisfying the requirements of this component provides (1) fail-safe defaults (i.e., defaults that deny access whenever a user or administrator fails to specify access to subjects and objects), (2) user-defined defaults, (3) well-defined interface conventions, (4) the users' ability to reduce their own privileges, and (5) subject, object, resource, and service protection in common configurations. Without this component, the protection value of the TCB functions is diminished since few users and applications would be able to employ these functions effectively.

## 4.2 TCB Functional Components

The TCB functional components are presented in terms of the generic threats and vulnerabilities they are intended to counter or eliminate. Most protection profiles for IT products based on operating systems will include most of the functional components presented in the following subsections. Protection profiles for other types of IT products may include only some of these components depending upon the product's purpose.

### 4.2.1 Security Policy Support

The focus of information protection within an IT product is to support an organization's security policies. This section describes TCB functions and associated components (i.e., accountability, access control, availability, and security management) that help support organizational security policies. The generic functional components have been written to be *policy neutral* (i.e., they are capable of supporting a wide variety of protection policies). Specific product policies or types of product policies (e.g., policies derived from the DoD policy for confidentiality, a hospital's policy for privacy and integrity of patient records, or a phone company's policy for availability) can be defined by assigning a specific meaning to, or refining the generic, policy neutral components. Details of profile construction and synthesis of profile components from generic components are provided in Chapter 7.

#### 4.2.1.1 Accountability Policy

An IT product that supports accountability policies must include functions capable of attributing responsibility for an action to an accountable entity (i.e., the identified and authenticated individual whose policy attributes may include name, role, group, and/or security level). Accountability requirements may be satisfied in a product through the use of the following functional components. *Identification and authentication* components establish the authenticity of the claimed identity by the user. *System entry* components provide the appropriate time, location, and mode-of-entry context for the user's interactions. *Trusted path* components ensure that nothing can interfere with the interactions between the TCB and the authenticated user. *Audit* components ensure that user interactions are recorded and attributed to the accountable user identity. Each of these components is discussed in more detail in the following subsections.

##### 4.2.1.1.1 Identification & Authentication (I&A)

I&A components specify functional requirements for the TCB to verify the claimed identity of individuals attempting system entry. Identification and authentication is required to ensure that the authenticated users are associated with the proper set of policy attributes (e.g., identity, groups, roles, security or integrity levels, time intervals, location). Thus, identification and authentication enables the TCB to ensure that all individuals entering a system and accessing its subjects, objects, and services are authorized to do so by the system entry and TCB's protection policy, and that the accountability policy can be enforced. In operating systems, the I&A functions constitute the main part of the process commonly known as "login," with the balance of the process consisting of system entry and trusted path functions.

#### 4.2.1.1.2 System Entry

The system entry components specify functional requirements for the control of an identified and authenticated user's entry into the system. The user's entry into the system typically consists of the creation of one or more subjects that execute instructions in the system on behalf of the user. At the end of the system entry procedure, provided the system entry conditions are satisfied, the created subjects bear the policy attributes determined by the I&A functions. System entry conditions can be specified in terms of policy attributes such as the user's identity, group or role membership, confidentiality and integrity levels, time intervals, location, and mode of access.

The system entry procedure may include warnings about unauthorized attempts to gain access to the system. It may also display last login data to the user, so that the user can determine whether the previous successful login was performed by the user and not by an intruder who successfully broke the user's password, for instance. The system entry procedure may enable the control of (1) multiple simultaneous user logins, (2) locking an interactive session during periods of user inactivity, (3) time intervals during authorized user access, and (4) location or port of user entry.

System entry control can help counter threats of inadvertent, deliberate, or coerced access performed in an unauthorized manner by an authenticated user. For example, the location and time of system entry can be constrained in such a way that identified and authenticated users located in areas of high exposure (e.g., public areas) cannot display sensitive data, enter high-integrity commands, or operate outside working hours. Similarly, controlling the mode of system entry helps ensure that identified and authenticated users cannot remotely start batch computations that would normally require the user's attendance.

#### 4.2.1.1.3 Trusted Path

Trusted path components specify functional requirements for ensuring that users have direct, unencumbered communication with the TCB. A trusted path may be required at login time and at other times during a subject session. These trusted path exchanges may be initiated by a user during a TCB interaction. However, a TCB or a trusted application request for user input should also allow a user to initiate and respond via the trusted path. A user's response via the trusted path guarantees that untrusted applications cannot intercept and/or modify the user's response.

The threats countered by these components are unauthorized discovery and/or modification of user-private information associated with commands (e.g., login password, sensitivity of the user's actions), and modification of commands and command parameters causing incorrect user input to the TCB. Trusted path programs of the TCB may also be invoked by trusted applications to ensure correct display of information to the user. These programs may also allow the addition of trusted application commands to the trusted path so that users could communicate securely with these applications.

Absence of a trusted path may allow breaches of accountability in environments where untrusted applications are used. These applications can intercept user-private information, such as passwords, and use it to impersonate other legitimate users. As a consequence, responsibility for any system actions cannot be reliably assigned to an accountable entity. Also, these applications could output erroneous information on an unsuspecting user's display. Thus, subsequent user actions may be erroneous and may lead to security breaches.

#### 4.2.1.1.4 Audit

The audit components specify requirements for monitoring and, in some cases, detecting real or potential violations of security policies in organizations that use IT products containing audit functions. These functions help monitor the use of access rights by authorized users, and act as a deterrent against usage policy violations.

Auditing involves recognizing, recording, and analyzing user actions that are considered, by audit administrators, to be critical to the success of an organization's security policy. The resulting audit records can be examined to determine which security-relevant user actions took place and who was responsible for them. The audit component requirements refer to the basic audit mechanisms, including audit data protection, record format and event selection, as well as to analysis tools, violation alarms, and real-time intrusion detection systems, which use the basic mechanisms.

Recognition of auditable actions is based largely on administratively supplied specifications of user actions and patterns of behavior whose appropriateness is considered to be significant to the satisfaction of an organization's security policy. The designers of an IT product must either anticipate which actions and patterns are likely to be considered important to organizations with respect to their security policies, or provide an audit interface that allows trusted (and possibly other) applications to recognize and pass audit data to the TCB. Since the purpose of the audit mechanism is to audit user actions, including administrative actions, designers of the audit mechanism cannot uniformly assume that all authorized actions are appropriate; consequently. some administrative actions must always be audited.

The IT product must record each action that has been deemed auditable along with accompanying information needed to understand the apparent purpose or effect of that action (e.g., user environment variables, programs used to preprocess user input). Recorded audit data must be protected by the TCB from inappropriate modification, use, or destruction. To avoid repudiation, the mechanism by which audit data is gathered must be known and reliable. Often this implies the use of a trusted communications mechanism. At higher levels of assurance, the auditing of key administrative actions should resist all attacks by remote users and otherwise undetectable attacks by users with access to the physical audit media (e.g., through the use of write-once audit disks).

Finally, audit data must be available for analysis in a timely manner and in a useful format, within policy constraints established for the product. This requirement motivates the design of pre- and post-processing software that organizes audit data into a presentable format and/or delivers it to authorized users or processes acting on their behalf.

#### 4.2.1.2 Access Control Policy

The access control objectives of organizational security policies can be divided into two classes, namely *confidentiality* and *integrity***.** These objectives determine whether the organization intends to prevent unauthorized disclosure or unauthorized modification and destruction of information. Often, organizational security policies include both confidentiality and integrity objectives to varying degrees. These policies reflect both security and system management goals that should be satisfied by multiple IT products.

The extent to which an IT product's access control policy supports high-level system and organizational security policy objectives varies from product to product. Few commercial products are designed to support a single specific organizational policy. Instead, commercial products implement either low-level access control policies that can be tailored to support high-level organizational policies or multiple organizational policies that could be individually instantiated on a system basis. For example, some products implement both the DoD mandatory confidentiality policy (as modeled by Bell & LaPadula) and a mandatory integrity policy (as modeled by Biba). When using such IT products in environments where only the mandatory integrity policy needs to be enforced, the DoD mandatory confidentiality policy could be deconfigured (e.g., all authorization checks for DoD mandatory confidentiality would pass and all options for displaying, or requesting, confidentiality levels would be disabled). Similarly, other organizational policies, such as the role-based access control policies, could be configured in a product when the environment of product use makes it necessary.

The access control policies in this section are IT product policies implemented by TCB functional components and are distinguished from the higher level system and organizational security policies, which generally use product policies to help achieve the higher level security objectives. Product access control policies are designed to counter generic threats. These policies traditionally have been classified as *discretionary* or *non-discretionary*, depending upon whether the access control decisions regarding an object are primarily based on actions of the unprivileged user and/or subject that created the object or primarily based on administrative actions. Access control policies of many products combine both discretionary and non-discretionary policies to counter different types of threats and eliminate various vulnerabilities.

#### 4.2.1.2.1 Discretionary Access Control Policies

The generic threats addressed by discretionary access control policies are those of unauthorized access, propagation or retention of access rights to user's objects, and unauthorized creation or destruction of subjects and objects. Discretionary access controls enable users and applications to protect their objects from unauthorized access by other users and applications. These controls are effective, provided that malicious code is not introduced and used by a user or on behalf of a user.

Discretionary access control policies cannot counter and are not intended to address several generic threats and vulnerabilities such as Trojan horse or virus propagation within a user application. This is because these policies have traditionally imposed very few restrictions on object sharing. Most commercially available IT products that support only discretionary policies could not adequately control or confine the actions of a Trojan horse or a virus within an application. Furthermore, discretionary policies are not intended to control the flow of information between a subject and/or object to system variables that do not represent subjects and/or objects (e.g., internal variables of an operating system). Consequently, the use of covert channels is a threat that cannot be countered by any discretionary access control policy.

Discretionary access controls are also not intended to prevent surrogate access to objects. As a typical example of surrogate access, consider an object's owner who allows user A, but not user B, to read the contents of one of the owner's objects. However, the object owner cannot exercise any control over user A's discretion on how to use the object contents. User A can transfer the contents of the owner's object to user B in an authorized manner via the interprocess communication facilities; or user A may simply copy the contents of the owner's object into another object shared with user B. The object owner cannot control user A's legitimate discretionary communications with user B, and thus, the object owner cannot control the flow of data to and from the object caused by user A on behalf of user B.

A range of discretionary policies have been used by various IT products to satisfy different protection requirements. These policies range from those where the owner (creator or controller) of an object (and an application running on the owner's behalf) has complete control over who can access that object to those where any possessor of an access right to an object can freely distribute that access right to, and subsequently revoke it from, other users and applications.

Generic threats to access control not countered by discretionary access controls are intended to be countered by non-discretionary access controls. These non-discretionary access control policies are discussed in the next section.

#### 4.2.1.2.2 Non-Discretionary Access Control Policies

Non-discretionary access control policies are intended to counter threats posed by malicious code (e.g., Trojan horses or virus codes) within application programs, by surrogate subjects, and in general, to counter both unauthorized access to objects and unauthorized flow of information between subjects and objects, not just unauthorized propagation of access rights. An IT product that provides non-discretionary access control can confine the effects of malicious code and the flow of information between subjects and objects as specified by system administrators. In general, non-discretionary controls are specified by security administrators and cannot be changed over time by unprivileged subjects. Thus, the unprivileged subject's discretion as to whether an object can be accessed is limited by administrative controls. Also, an unprivileged user can only exercise very limited access-control discretion. By selecting certain policy attributes from the attribute sets defined by administrators (e.g., role, session security level), the user selects the access control attributes for subjects created for him/her to run external to the TCB. Non-discretionary policies allow the basis for determining whether a subject could have access to an object based exclusively on the subject's and the object's non-discretionary policy attributes. In this sense, non-discretionary access controls can confine user and application program activity.

Unlike discretionary access controls, which typically do not offer separate and explicit support for specific confidentiality and integrity policies beyond distinguishing between attributes for reading and writing objects, non-discretionary controls can demonstrate support for high-level organizational policies. This is due, in part, to the central (organizational) role played by system administrators in the control of access authorization and object sharing, as opposed to discretionary policies where individual object creators, not system administrators, play this access authorization and object-sharing-control role.

Various non-discretionary access control policies have been used in different products. These policies range from the DoD mandatory policies used to protect the confidentiality of classified documents to separation of role and separation of duty policies intended to protect the integrity of databases. Also, some products have a capability to enforce both non-discretionary confidentiality and integrity controls on the same or different sets of subjects and objects.

Both non-discretionary confidentiality and integrity policies may, or may not, adequately control the flow of information and the use of covert channels. Not all non-discretionary policies are aimed at controlling the use of covert channels. Should covert channels be considered a threat, however, both non-discretionary confidentiality and integrity policies require measures of covert channel handling. These measures are discussed in the next section.

#### 4.2.1.2.3 Covert Channel Handling

Covert-channel handling components include both technical requirements (e.g., elimination, bandwidth reduction to acceptable levels, deterrence of use by auditing covert storage channels), and administrative or environmental requirements (e.g., exclusive use of trusted software by trusted users in environments where all unauthorized information flow must be prevented).

Covert-channel elimination requires that the design and/or implementation of a system be changed so that covert channels are removed from the product. These changes include (1) the elimination of resource sharing between any subjects that could take part in covert channel use by preallocating maximum resource demands to all such subjects or by partitioning resources on a per-subject basis, and (2) the elimination of interfaces, features, and mechanisms which can cause covert leakage. Since covert-channel elimination may be impractical for some channels, other handling functions may be useful in a TCB (e.g., bandwidth limitation functions).

Covert-channel bandwidth limitation requires that the maximum, or alternatively, the average bandwidth of any channel be reduced to a limit deemed acceptable in the environment of product use. In sensitive applications, bandwidth limitation may require that the aggregated (i.e., combined) bandwidth of a product's covert channels be reduced to an acceptable value. Bandwidths can be limited by (1) deliberate introduction of noise in TCB functions used to exploit the channels (e.g., use of random allocation algorithms for shared resources such as indexes in shared tables, disk areas, and process identifiers, or introduction of extraneous processes that modify covert channel variables of a TCB in pseudo-random patterns), or (2) deliberate introduction of delays in each TCB primitive of a real channel.

Covert-channel auditing is a primary method used to discourage the use of covert channels. This method assumes that the frequent use of a channel can be unambiguously detected by audit mechanisms. Some covert channels preclude the use of channel audit, elimination, and bandwidth limitation methods. These channels typically include the timing channels that arise from hardware-resource sharing (e.g., shared busses, processor caches). Furthermore, in some environments, the threat analysis may indicate that any use of covert channels cannot be tolerated. However, in most commercial products it is impractical to eliminate all covert channels. If such products are used in such non-tolerant environments, the effect of covert-channel use must be neutralized. This could be done by the exclusive use of trusted product and application software. In such cases, evidence must be provided to justify that the exclusive use of trusted application software is sufficient to render the existing covert channels ineffective.

### 4.2.1.3 Availability Policy

An IT product which supports availability policies must provide protection functions capable of controlling the availability of the product subjects, objects, resources, and services. Availability components refer to policies for prevention, detection, and recovery from unauthorized denial of service caused by unprivileged subjects. These components also refer to the use of redundancy and recovery from lack of availability caused by TCB failures. Because subjects and objects are represented by, and consume, system resources such as primary memory and disk space, CPU time, and shared TCB internal tables and objects, the allocation of these resources must be controlled to allow policy-ensured accesses to take place.

A product that controls the availability of subjects, objects, and services may include TCB functions that prevent denial of service and provide fault tolerance. The needed availability functions of a TCB may include resource allocation containment and fault tolerant services.

#### 4.2.1.3.1 Resource Allocation

Resource allocation functions allow the TCB to control the use of product resources by users and subjects such that denial of service will not take place. Denial-of-service protection can be provided by containing resource allocations in time and space, or by establishing priority-based allocations.

Resource allocation rules may allow the creation of quotas or other means of defining limits on the amount of resource space or time that may be allocated on behalf of a specific user, process, or task. These rules may provide for object quotas that constrain the number and/or size of objects a specific user may allocate. Resource allocation rules may control the allocation/deallocation of pre-assigned resource blocks where these blocks are defined under the control of the TCB. Under these rules, subjects and objects are assigned allocation attributes so that the TCB can enforce appropriate quotas. Finally, resource allocation rules may prioritize subject access to resources so that subjects with the highest priorities are given preferential access to these resources.

#### 4.2.1.3.2 Fault Tolerance

TBD.

#### 4.2.1.4 Security Management

The TCB of an IT product must support security management components to enable administrative users to set up and control the secure operation of the product. These components refer to TCB functions associated with both administrator and operator roles, and have both access control, audit, and availability relevance.

Security management components refer to the following types of functions:

a. TCB generation, installation, configuration, and non-routine maintenance (e.g., TCB manual recovery, installation of "patches" correcting security flaws, repair of damaged TCB hardware and software elements).

b. Definition and update of user security characteristics (e.g., unique identifiers associated with user names, user accounts, per-user policy attributes, system entry parameters, availability parameters or resource quotas).

c. Definition and update of security policy parameters (e.g., identification and authentication, system entry, access control, and availability parameters).

d. Routine control and maintenance of product resources (e.g., enable and disable peripheral devices, mounting of removable storage media, backup and recovery of user objects, and routine maintenance of TCB hardware and software elements).

e. Auditing both privileged and unprivileged user actions, and audit management (e.g., selection of audit events, management of audit trails, audit trail analysis, and audit report generation).

The security management functions help counter the same threats as those countered by the security policy functions (i.e., accountability, access control, and availability). This is the case because the security management functions implement a significant part of all the system security policies. In addition, when the security management functions are partitioned into different administrative roles, they help limit the potential damage caused by unskilled or corrupt administrators.

### 4.2.2 Reference Mediation

Functions that implement a security policy provide effective protection against unauthorized access only if all references (i.e., denoted by <action; object(s) > tuples) issued by subjects are directed by the TCB code to the appropriate security policy modules for validation. Should such references be incorrectly directed, or not directed at all, to the required policy modules, policy enforcement will be incorrect, incomplete, or absent, despite correct and complete policy implementation. This would allow unprivileged subjects to bypass security policy in a variety of unauthorized ways (e.g., bypass certain access checks for a subset of the objects and subjects, bypass all checks for a type of object whose protection was assumed by applications, retain access rights beyond their intended expiration time, and/or bypass audit).

Note that the requirements of the reference mediation component are independent of the particular policies supported by a product.

### 4.2.3 TCB Logical Protection

The protection of the TCB from external interference and tampering is a fundamental component of any secure product. Should unprivileged subjects read or modify TCB elements (i.e., data structures and code), the security policy might be circumvented or even modified in potentially undetectable ways.

The reading of TCB internal variables, that is, variables that are not part of any defined subject or object (e.g., internal TCB buffers, table entries), would not be addressed by low-level product policies defined solely in terms of subjects and objects. In this case, reading by users or subjects outside the TCB would not be prohibited, even though it could result in failure to support the organizational policies. Similarly, modification of TCB internal variables may cause (1) the introduction of miscreant code into the TCB, which can modify product policies, (2) the modification of user and application-level objects that depend on the consistency of the TCB internal variables, (3) denial of service to users and applications, and/or (4) covert transfer of information through the TCB in violation of information-flow policy. Unauthorized acquisition of privileges might allow the reading and modification of TCB internal variables and objects (e.g., password files, group and/or role definition files, files defining security and/or integrity levels) and might allow unprivileged users to execute privileged functions.

To provide TCB isolation, all references to TCB internal entities and all access rights passed by unprivileged subjects to the TCB must be mediated in a non-circumventable manner. This particular form of mediation is not specified as an access mediation requirement because a cyclic dependency would be introduced between access mediation and TCB protection. This is the case because the correct reference mediation depends on TCB protection (see discussion in Chapter 7, “Construction of Protection Profiles”).

### 4.2.4 TCB Physical Protection

TCB physical protection components refer to restrictions of unauthorized physical access to the TCB, and to deterrence of unauthorized physical use, modification, or substitution of the TCB.

### 4.2.5 TCB Self-Checking

TCB self-checking functions are needed to detect the corruption of protection-relevant code and data structures by various failures that do not necessarily stop the product's operation (which would be handled by TCB recoverability). These checks must be performed because these failures may not necessarily be prevented. Such failures can occur either because of unforeseen failure modes and associated oversights in the design of hardware, firmware, or software, or because of malicious corruption of the TCB due to inadequate physical TCB protection.

### 4.2.6 TCB Start-Up and Recovery

TCB recovery components refer to the functions that respond to anticipated failures or discontinuity of operations. These functional components cannot handle "unanticipated" failures or discontinuity of operation, and manual administrative procedures must be employed for such events.

Recovery components reconstruct TCB secure states or prevent transitions to insecure states as a direct response to occurrences of expected failures, discontinuity of operation or start-up. Failures that must be generally anticipated include (1) actions failures (e.g., actions that fail to complete because they detect exceptional conditions during their operation); (2) unmaskable action failures that always cause a system crash (e.g., persistent inconsistency of critical system tables, uncontrolled transfers within TCB code caused by transient failures of hardware or firmware, power failures, processor failures); (3) non-volatile media failures causing part or all of the media representing TCB objects to become inaccessible or corrupt (e.g., disk head crash, persistent read/write failure caused by misaligned disk heads, worn-out magnetic coating, dust on the disk surface); and (4) discontinuity of operation caused by erroneous administrative action or lack of timely administrative action (e.g., unexpected shutdowns by turning off power, ignoring the exhaustion of critical resources, inadequate installed configuration).

### 4.2.7 TCB Privileged Operation

Functions that limit the privileges available to the TCB are primarily intended to limit the damage that can be caused by errors and failures of TCB mechanisms. To accomplish this, it is necessary to limit the interactions among privileged TCB components to a minimum such that improper use of privileges by a TCB function, module, or action as a consequence of failures or accidents will have limited or no effect on other components. For example, the association of privileges with different administrative commands facilitates the separation of administrative roles. Similarly, the association of different privileges with TCB components that have no functional interaction, such as audit trail and password management components, limits the possibility of unwarranted interaction. As a consequence, if a penetration of a component takes place, the likelihood that other unrelated components are also penetrated may be diminished. The finer the granularity of privileges and of privilege association with TCB functional components, actions of components, and administrative roles, the less chance of damage caused by errors, failures, accidents, and penetrations.

### 4.2.8 TCB Ease-of-Use

The notion that an IT product must include functions which facilitate and enhance the use of basic protection mechanisms is motivated by two related observations. First, if a product’s protection mechanisms are complex, difficult to use, or have inadequate performance, they will not be used by system administrators or by application programmers. The mere presence of (potentially elaborate) security policies in a product is insufficient to facilitate the development or use of secure applications and the secure management of a product. An IT product may still be vulnerable to inadvertent errors caused by difficulties in using the product’s protection functions. Second, functions that facilitate and enhance the use of basic protection mechanisms may be difficult to retrofit into a product because of their pervasiveness. Instead, to be effective, these components must be included in the initial product design.

### 4.3 Rated Functional Components

Functional components can be selected for inclusion in a profile based on environment-specific requirements (see Chapter 3). To facilitate this selection and compatibility with existing criteria, each of the functional components of a TCB is *rated*. The rating of the TCB functional components is based on the following four parameters: (1) the *scope* of the requirement application, (2) the *granularity* of the requirement, (3) the *coverage* of a requirement's features, and (4) the *strength* of the requirement.

**Scope.** The scope of a requirement determines the *entity subset* to which the requirement applies; i.e., (1) to all the users, subjects and objects, (2) to all the TCB functions and application programming interfaces, (3) to all TCB elements (i.e., hardware, firmware, software, data structures and code), and (4) to all TCB configurations, or only to a defined subset thereof. For example, the access control, audit, availability, reference mediation, and ease-of-use components may refer only to certain subsets of objects and configurations; trusted path may include only certain subsets of the TCB commands (only login commands but not change-of-password commands or change-role commands); and the recovered secure state of the TCB may include all the user objects or only a defined set.

**Granularity.** The granularity of a requirement determines the *entity-attribute subset* to which the requirement applies (e.g., whether the requirement applies to all attributes of users, subjects or objects, or only to a defined subset of attributes). Access control, audit, and reference mediation may include only certain attributes of subjects and objects, but not others. For example, access control, audit, and reference mediation may refer to access rights for objects and subjects, but not to object and subject status variables; authentication may be based on group or role identities, but not on individual user identity; privileges may be associated with roles, but not with individual TCB functions or actions (e.g., function invocations).

**Coverage.** The coverage of a requirement determines the *feature subset* included in that requirement. This is illustrated in the following examples:

a. Access control may include only discretionary features of authorization, administration of policy attributes (e.g., user identities, groups, security and/or integrity levels, roles), and object and/or subject creation and destruction, but not encapsulation.

b. Audit may include only post-processing analysis tools for detecting accumulation of events (e.g., multiple failed logins) but not real-time alarms.

c. Availability may include resource restrictions but not prioritized resource allocation.

d. TCB protection may include only isolation features but not TCB consistency features.

e. Physical TCB protection may include only attack detection and deterrence features, but not attack countermeasures.

f. TCB self-checking may be periodical or continuous.

g. Recovery may be only manual, not automatic.

h. The ease-of-use mechanisms may include administrative and application programming support features but may not minimize performance penalties of using them.

**Strength.** The strength of a requirement supported by a function defines the *conditions* under which that function withstands a defined attack or tolerates failures. For example, the user authentication function may withstand certain kinds of impersonation attacks but not others (e.g., the password complexity rules may counter human guessing attacks but not automated attacks using a dictionary). Other examples include conditions in which conjunction of independent user authentication mechanisms yields stronger authentication than the use of either mechanism alone, or a certain encryption mechanism for one-way function computation may have different work factor characteristics than other encryption mechanisms. Similarly, the TCB physical protection characteristics may vary according to different work factor characteristics.

The strength of a requirement may also be used to differentiate access control policies. For example, non-discretionary access controls are typically stronger than discretionary access controls with respect to their ability to counter attacks mounted by miscreant application code executing programs on behalf of an unsuspecting user. However, this notion of strength is not used to rate individual access control components. Instead, it is used in the analysis of the protection profiles (i.e., in assessing whether a chosen access control policy can counter specific threats).

Rating implies some form of ordering. The independent application of the scope, granularity, coverage, and strength parameters to distinguish between the levels of functional components may not necessarily lead to a linear ordering among these levels. To obtain such an ordering these rating parameters are applied in the order in which they are listed above. Whenever all rating parameters apply to a given functional component, lower levels are distinguished by scope and granularity and higher levels by coverage and strength. However, this ordering of the rating parameters does not imply that each component level represents a component extension resulting from the application of a *single* rating parameter. Instead, a component level change may represent a component extension resulting from the application of *several* rating parameters characterizing the intent of a functional component (e.g., support of a specific policy, compatibility with existing standards and guidelines).

The above parameters and ordering are chosen to enable the rating of functional components at levels of detail comparable to those of existing standards (e.g., TCSEC, CTCPEC, ITSEC), thereby enabling potential harmonization with these standards. However, the rating of functional components does not restrict a profile developer to the choices of rated components presented. As illustrated in Chapter 7, a profile developer can synthesize new components from existing ones (e.g., by assigning a specific meaning to a generic requirement, by refining a requirement of a component, by augmenting a lower rated component with an individual requirement of a higher rated component) within the constraints of dependency analysis.

The means of rating each component are summarized in Table 2.

**Table 2. Rated Functional Components**

| Functional Component | Scope | Granularity | Coverage | Strength |
|---|---|---|---|---|
| Security Policy Support | | | | |
| Accountability | | | | |
| Identification & Authentication | | | x | x |
| System Entry | | | x | |
| Trusted Path | x | | x | |
| Audit | | | x | x |
| Access Control | x | x | x | |
| Covert Channel Handling | x | | x | |
| Availability | | | | |
| Resource Allocation | x | | x | |
| Fault Tolerance (TBD) | --- | --- | --- | --- |
| Security Mgmt. | | | x | x |
| Reference Mediation | x | x | x | |
| TCB Logical Protection | | | x | |
| TCB Physical Protection | | | x | x |

**Table 2. Rated Functional Components**

| Functional Component | Scope | Granularity | Coverage | Strength |
|---|---|---|---|---|
| TCB Self-checking | x | | x | |
| TCB Start-Up and Recovery | | | x | |
| TCB Privileged Operation | | x | | |
| TCB Ease-of-Use | x | | x | |

### 4.3.1 Rated Identification & Authentication Components

Identification and authentication is a required component for most security policies. Without this component, the threat of unauthorized or inappropriate system entry and access to resources could not be countered. However, weak identification and authentication functions could not counter the threat of impersonation attacks by unauthorized users. For this reason, identification and authentication components are noted based on both the coverage and strength of the authentication features. Furthermore, the combined use of more than one type of authentication can provide greater control over unauthorized access.

The features covered at level I&A-1 include only minimal forms of individual user authentication. This level of I&A is intended for use in products with limited capabilities, such as automated guards, where basic I&A and system-entry audit are the primary functions supported. In contrast, the features of level I&A-2 include policy attributes that are determined on an individual basis, thereby providing basic authorization. The use of this level is anticipated in most operating systems where policy attributes, such as groups and security levels, need to be authenticated for system entry. Level I&A-3 extends the feature coverage of level I&A-2 by providing a well-defined set of responses to authentication exceptions and a capability to maintain, protect and display user status information. The use of this level is anticipated to include products with well-defined access control and availability policies as well as system-entry control. The level I&A-4 extends the feature coverage of level I&A-3 by requiring that installable mechanisms be supported, and that a policy be enforced that assigns a specific authentication procedure to each user, or to a policy attribute of each user. Level I&A-5 strengthens level I&A-4 by requiring that two or more identification and authentication mechanisms authenticate certain user identities or other policy attributes.

**I&A-1 Minimal Identification and Authentication**

**1. The TCB shall require users to identify themselves to it before beginning to perform any other actions that the TCB is expected to mediate. The TCB shall be able to enforce individual accountability by providing the capability to uniquely identify each individual user. The TCB shall also provide the capability of associating this identity with all auditable actions taken by that individual.**

**2. The TCB shall use a protected mechanism (e.g., passwords) to authenticate the user's identity.**

**3. The TCB shall protect authentication data so that it cannot be used by any unauthorized user.**

**I&A-2 Identification, Authentication, and Authorization**

1. The TCB shall require users to identify themselves to it before beginning to perform any other actions that the TCB is expected to mediate. The TCB shall be able to enforce individual accountability by providing the capability to uniquely identify each individual user. The TCB shall also provide the capability of associating this identity with all auditable actions taken by that individual.

2. **The TCB shall maintain authentication data that includes information for verifying the identity of individual users (e.g., passwords) as well as information for determining the product policy attributes of individual users (e.g., groups, roles, security and/or integrity levels, time intervals, location). These data shall be used by the TCB** to authenticate the user's identity **and to ensure that the attributes of subjects external to the TCB that may be created to act on behalf of the individual user satisfy the product policy (e.g., the subject security level and authorizations are dominated by the clearance and authorization of that user).**

3. The TCB shall protect authentication data so that it cannot be used by any unauthorized user.

**I&A-3 Exception-Controlled Identification and Authentication**

1. The TCB shall require users to identify themselves to it before beginning to perform any other actions that the TCB is expected to mediate. The TCB shall be able to enforce individual accountability by providing the capability to uniquely identify each individual user. The TCB shall also provide the capability of associating this identity with all auditable actions taken by that individual.

2. The TCB shall maintain authentication data that includes information for verifying the identity of individual users (e.g., passwords) as well as information for determining the product policy attributes of individual users (e.g., groups, roles, security and/or integrity levels, time intervals, location). These data shall be used by the TCB to authenticate the user's

identity and to ensure that the attributes of subjects external to the TCB that may be created to act on behalf of the individual user satisfy the product policy (e.g., the subject security level and authorizations are dominated by the clearance and authorization of that user).

3. The TCB shall protect authentication data so that it cannot be used by any unauthorized user. **The TCB shall appear to perform the entire user authentication procedure even if the user identification entered is invalid.**

**The TCB shall end the attempted login session if the user performs the authentication procedure incorrectly for a number of successive times (i.e., a threshold) specified by an authorized system administrator. A default threshold shall be defined. When the threshold is exceeded, the TCB shall send an alarm message to the system console and/or to the administrator's terminal, log this event in the audit trail, and delay the next login by an interval of time specified by the authorized system administrator. A default time interval shall be defined. The TCB shall provide a protected mechanism to disable the user identity or account when the threshold of successive, unsuccessful login attempts is violated more than a number of times specified by the administrator. By default, this mechanism shall be disabled (as it may cause unauthorized denial of service).**

4. **The TCB shall have the capability to maintain, protect, and display status information for all active users (e.g., users currently logged on, current policy attributes) and of all user accounts (i.e., enabled or disabled user identity or account).**

### I&A-4 Installable I&A Mechanisms

1. The TCB shall require users to identify themselves to it before beginning to perform any other actions that the TCB is expected to mediate. The TCB shall be able to enforce individual accountability by providing the capability to uniquely identify each individual user. The TCB shall also provide the capability of associating this identity with all auditable actions taken by that individual. **Furthermore, the TCB shall have the capability of associating a unique identity with each privileged subject.**

2. The TCB shall maintain authentication data that includes information for verifying the identity of individual users (e.g., passwords) as well as information for determining the product policy attributes of individual users (e.g., groups, roles, security and/or integrity levels, time intervals, location). These data shall be used by the TCB to authenticate the user’s identity and to ensure that the attributes of subjects external to the TCB that may be created to act on behalf of the individual user satisfy the product policy (e.g., the subject security level and authorizations are dominated by the clearance and authorization of that user).

**The TCB shall be able to incorporate and use installable authentication mechanisms, such as token-based cards, biometrics, or trusted third-party mechanisms, in the place of or in addition to the default authentication (e.g., password-based) mechanism, to authenticate the user. The TCB shall be able to enforce separate user authentication procedures based on specific policy attributes.**

3. The TCB shall protect authentication data so that it cannot be used by any unauthorized user. The TCB shall appear to perform the entire user authentication procedure even if the user identification entered is invalid.

The TCB shall end the attempted login session if the user performs the authentication procedure incorrectly for a number of successive times (i.e., a threshold) specified by an authorized system administrator. A default threshold shall be defined. When the threshold is exceeded, the TCB shall send an alarm message to the system console and/or to the administrator's terminal, log this event in the audit trail, and delay the next login by an interval of time specified by the authorized system administrator. A default time interval shall be defined. The TCB shall provide a protected mechanism to disable the user identity or account when the threshold of successive, unsuccessful login attempts is violated more than a number of times specified by the administrator. By default, this mechanism shall be disabled (as it may cause unauthorized denial of service).

4. The TCB shall have the capability to maintain, protect, and display status information for all active users (e.g., users currently logged on, current policy attributes) and of all user accounts (i.e., enabled or disabled user identity or account).

**I&A-5 Multiple I&A Mechanisms**

1. The TCB shall require users to identify themselves to it before beginning to perform any other actions that the TCB is expected to mediate. The TCB shall be able to enforce individual accountability by providing the capability to uniquely identify each individual user. The TCB shall also provide the capability of associating this identity with all auditable actions taken by that individual. Furthermore, the TCB shall have the capability of associating a unique identity with each privileged subject.

2. The TCB shall maintain authentication data that includes information for verifying the identity of individual users (e.g., passwords) as well as information for determining the product policy attributes of individual users (e.g., groups, roles, security and/or integrity levels, time intervals, location). These data shall be used by the TCB to authenticate the user's identity and to ensure that the attributes of subjects external to the TCB that may be created to act on behalf of the individual user satisfy the product policy (e.g., the subject security level and authorizations are dominated by the clearance and authorization of that user).

The TCB shall be able to incorporate and use installable authentication mechanisms, such as token-based cards, biometrics, or trusted third-party mechanisms, in the place of or in addition to the default authentication (e.g., password-based) mechanism, to authenticate the user. The TCB shall be able to enforce separate user authentication procedures based on specific policy attributes. **Each user shall be authenticated by two or more types of authentication mechanisms; i.e., the authentication is successful only if all mechanisms individually indicate successful authentication. The TCB shall be able to enforce the use of these mechanisms on a policy-attribute basis.**

3. The TCB shall protect authentication data so that it cannot be used by any unauthorized user. The TCB shall appear to perform the entire user authentication procedure even if the user identification entered is invalid.

The TCB shall end the attempted login session if the user performs the authentication procedure incorrectly for a number of successive times (i.e., a threshold) specified by an authorized system administrator. A default threshold shall be defined. When the threshold is exceeded, the TCB shall send an alarm message to the system console and/or to the administrator's terminal, log this event in the audit trail, and delay the next login by an interval of time specified by the authorized system administrator. A default time interval shall be defined. The TCB shall provide a protected mechanism to disable the user identity or account when the threshold of successive, unsuccessful login attempts is violated more than a number of times specified by the administrator. By default, this mechanism shall be disabled (as it may cause unauthorized denial of service).

4. The TCB shall have the capability to maintain, protect, and display status information for all active users (e.g., users currently logged on, current policy attributes) and of all user accounts (i.e., enabled or disabled user identity or account).

### 4.3.2 Rated System Entry Components

System entry control helps enhance accountability by providing a time, space, and mode-of-entry context to each action for which the user is held accountable. The additional constraints of system entry control help gain increased confidence that the proper user is held responsible for a set of authorized actions.

System entry by an identified and authenticated user shall be controlled by the TCB. The conditions under which a user subject (e.g., process) is created on behalf of an identified and authenticated user shall be specified. The specification of these conditions shall be based on users' policy attributes (e.g., groups, roles, security and/or integrity levels, time intervals, location).

The system-entry components are rated based on the coverage of specific conditions of system entry. For example, the features covered at level SE-1 include only basic forms of system entry (e.g., system entry conditions based on group or role membership, and security and/or integrity levels). This level is intended for use in most IT products that support system-entry control. Products that do not implement explicit system-entry control rely on the identification and authentication mechanism as the default system entry control. The features of level SE-2 include, in addition to the entry conditions of level SE-1, entry conditions defined in terms of the time and the location of entry. The level SE-3 extends the feature coverage of level SE-2 by requiring the explicit user ability to lock and unlock the user's own interactive sessions. Primitive forms of such locking by terminating and restarting a session are considered to have a substantially narrower coverage than those intended at this level and may be used only at lower levels.

### SE-1 Basic System Entry Control

**1. Prior to initiating the system login procedure, the TCB shall display an advisory warning message to the user regarding unauthorized use of the system and the possible consequences of failure to heed this warning.**

**2. Before system entry is granted to a user, the identity of that user shall be authenticated by the TCB. If the TCB is designed to support multiple login sessions per user identity, the TCB shall provide a protected mechanism to enable limiting the number of login sessions per user identity or account with a default of a single login session.**

**3. The TCB shall grant system entry only in accordance with the authenticated user's policy attributes. The system entry conditions shall be expressed in terms of users' policy attributes (e.g., greatest lower bound and least upper bound computations including the user levels, terminal levels, system levels). If no explicit system-entry conditions are defined, the system-entry default shall be used (e.g., the correct user authentication).**

**4. The TCB shall provide a protected mechanism that enables authorized administrators to display and modify the policy attributes used in system-entry control for each user. The conditions under which an unprivileged user may display these attributes shall be specified.**

**5. Upon a user's successful entry to the system, the TCB shall display the following data to the user and shall not remove them without user intervention: (1) the date, time, means of access and port of entry of the last successful entry to the system; and (2) the number of successive, unsuccessful attempts to access the system since the last successful entry by the identified user.**

**6. The TCB shall either lock or terminate an interactive session after an administrator-specified interval of user inactivity. The default value for this interval shall be specified.**

## SE-2 Time and Location Based Entry Control

1. Prior to initiating the system login procedure, the TCB shall display an advisory warning message to the user regarding unauthorized use of the system and the possible consequences of failure to heed this warning.

2. Before system entry is granted to a user, the identity of that user shall be authenticated by the TCB. If the TCB is designed to support multiple login sessions per user identity, the TCB shall provide a protected mechanism to enable limiting the number of login sessions per user identity or account with a default of a single login session.

3. The TCB shall grant system entry only in accordance with the authenticated user's policy attributes. The system entry conditions shall be expressed in terms of users' policy attributes (e.g., greatest lower bound and least upper bound computations including the user levels, terminal levels, system levels). If no explicit system-entry conditions are defined, the system-entry default shall be used (e.g., the correct user authentication). **The TCB shall provide a protected mechanism to allow or deny system entry based on specified ranges of time. Entry conditions using these ranges shall be specified using time-of-day, day-of-week, and calendar dates.**

   **The TCB shall provide a protected mechanism to allow or deny system entry based on location or port of entry. Conditions for system entry via dial-up lines (e.g., lists of user identities authorized to enter the system via dial-up lines), if any, shall be specified.**

4. The TCB shall provide a protected mechanism that enables authorized administrators to display and modify the policy attributes used in system-entry control for each user. The conditions under which an unprivileged user may display these attributes shall be specified.

5. Upon a user's successful entry to the system, the TCB shall display the following data to the user and shall not remove them without user intervention: (1) the date, time, means of access and port of entry of the last successful entry to the system; and (2) the number of successive, unsuccessful attempts to access the system since the last successful entry by the identified user.

6. The TCB shall either lock or terminate an interactive session after an administrator-specified interval of user inactivity. The default value for this interval shall be specified.

## SE-3 Session Locking and Unlocking

1. Prior to initiating the system login procedure, the TCB shall display an advisory warning message to the user regarding unauthorized use of the system and the possible consequences of failure to heed this warning.

2. Before system entry is granted to a user, the identity of that user shall be authenticated by the TCB. If the TCB is designed to support multiple login sessions per user identity, the TCB shall provide a protected mechanism to enable limiting the number of login sessions per user identity or account with a default of a single login session.

3. The TCB shall grant system entry only in accordance with the authenticated user's policy attributes. The system entry conditions shall be expressed in terms of users' policy attributes (e.g., greatest lower bound and least upper bound computations including the user levels, terminal levels, system levels). If no explicit system-entry conditions are defined, the system-entry default shall be used (e.g., the correct user authentication). The TCB shall provide a protected mechanism to allow or deny system entry based on specified ranges of time. Entry conditions using these ranges shall be specified using time-of-day, day-of-week, and calendar dates.

The TCB shall provide a protected mechanism to allow or deny system entry based on location or port of entry. Conditions for system entry via dial-up lines (e.g., lists of user identities authorized to enter the system via dial-up lines), if any, shall be specified.

4. The TCB shall provide a protected mechanism that enables authorized administrators to display and modify the policy attributes used in system-entry control for each user. The conditions under which an unprivileged user may display these attributes shall be specified.

5. Upon a user's successful entry to the system, the TCB shall display the following data to the user and shall not remove them without user intervention: (1) the date, time, means of access and port of entry of the last successful entry to the system; and (2) the number of successive, unsuccessful attempts to access the system since the last successful entry by the identified user.

6. The TCB shall either lock or terminate an interactive session after an administrator-specified interval of user inactivity. The default value for this interval shall be specified. **The TCB shall also provide a mechanism for user-initiated locking of the user's own interactive sessions (e.g., keyboard locking) that includes: (1) clearing or over-writing display devices to make the current contents unreadable; (2) requiring user authentication prior to unlocking the session; and (3) disabling any activity of the user's data entry/display devices other than unlocking the session.**

### 4.3.3 Rated Trusted Path Components

The trusted path components are rated based on the scope and coverage of the trusted-path interactions (e.g., user-TCB interactions including the number and types of commands included in the trusted path). Primitive forms of trusted path, such as terminating a login session or powering off a workstation to guarantee trusted path interaction, are considered to have a substantially narrower scope and coverage than those enabling trusted path within a login session.

The rating of the trusted path components intends to guarantee at the lowest level, TP-1, that a trusted communication channel exists from the user to the TCB for initial identification purposes. For higher levels, both the scope and the coverage of trusted path are extended. At level TP-2, trusted path includes not only login commands but also other commands that require protection (e.g., change of subject policy attributes). Thus, the TCB guarantees the invocation of a trusted communication channel from the user to the TCB for trusted sensitive commands and their parameters. At level TP-3, the coverage of the trusted path features is enlarged to enable trusted applications to communicate with the user for the validation of specific TCB mediated tasks (e.g., change of policy attributes, change of user registration parameters). This means that a trusted application can use a separate, trusted display feature, and that commands of the trusted application can be introduced in the user-initiated trusted path.

### TP-1 Login Trusted Path

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

### TP-2 Trusted User-to-TCB Communication

> The TCB shall support a trusted communication path between itself and users for **use whenever a positive user-to-TCB connection is required (e.g., login, change of policy attributes)**. Communications via this **trusted** path shall be **activated** exclusively by a user **or the TCB and shall be logically isolated and unmistakably distinguishable from other communication paths.**

### TP-3 Trusted Application-to-User Communication

> The TCB shall support a trusted communication path between itself and users for use whenever a positive user-to-TCB connection is required (e.g., login, change of subject or object attributes). Communications via this trusted path shall be activated exclusively by a user or the TCB and shall be logically isolated and unmistakably distinguishable from other communication paths.**The TCB shall also support a trusted communication path between trusted applications and users when a trusted application-to-user connection is required (e.g., display or input of application sensitive data).**

### 4.3.4 Rated Audit Components

The audit components are rated based on the coverage of the event-selection mechanisms and audit-analysis tools, and the strength of monitoring user actions (e.g., degree to which active, real-time monitoring is possible.) The audit requirements that follow are divided into four parts: first, the protection of the audit trail and the control of access to audit data; second, the definition of the auditable events; third, format and recording of the audit data; and fourth, the selection of audit events, and audit-data management, analysis, and reporting.

Level AD-1 includes minimal audit requirements; i.e., requirements that must be satisfied by all systems (to the extent to which they incorporate relevant policy functions). The audit coverage is extended at audit level AD-2 by extending the types of auditable events and by the inclusion of additional audit management functions. Audit function coverage is further extended at level AD-3 by the requirements for availability of trusted audit tools that enhance audit control (e.g., tools offering a graphical interface to the auditor, tools that enable the auditor to perform consistency checking of the selected events and of audit trails, tools that enhance the ease-of-auditing). Level AD-4 extends the coverage of the audit features of level AD-3 by requiring detection of accumulation of security-relevant events and generation of alarms whenever such events are detected. AD-5 represents an added level of auditing strength since it requires that auditing be performed in real-time. Thus, real-time intrusions can be detected.

**AD-1 - Minimal Audit**

1. **The TCB shall be able to create, maintain, and protect from modification or unauthorized access or destruction an audit trail of accesses to the objects it protects. The audit data shall be protected by the TCB so that read access to it is limited to those who are authorized for audit data.**

2. **The TCB shall be able to record the following types of events:**

   **- use of the identification and authentication mechanisms;**

   **- introduction of objects into a user's address space (e.g., file open, program initiation), and deletion of objects;**

   **- actions taken by computer operators and system administrators and/or system security officers.**

   **If availability policies are supported, attempts to circumvent or otherwise gain unauthorized access to resource-allocation limits shall be audited.**

**If non-discretionary access control policies are supported, the TCB shall be able to record any override of human-readable output markings. When the non-discretionary access control policies aim to control the flow of information between subjects, the TCB shall also be able to audit the identified event that may be used in the exploitation of covert channels.**

**3. 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 and policy attributes of the object (e.g., object security level).**

**4. The system administrator shall be able to selectively audit the actions of one or more users based on individual identity and/or object policy attributes (e.g., object security level).**

**AD-2 Basic Audit**

1. The TCB shall be able to create, maintain, and protect from modification or unauthorized access or destruction an audit trail of accesses to the objects it protects. The audit data shall be protected by the TCB so that read access to it is limited to those who are authorized for audit data.

2. The TCB shall be able to record the following types of events:

- use of the identification and authentication mechanisms**, and system entry events**;

- **access control events selectable on a per user, per subject, per object, and/or per policy attribute basis; i.e.,** introduction of objects into a user's address space (e.g., file open, program initiation), **creation and** deletion of **subjects and** objects; **distribution and revocation of access rights; changes of subject and object policy attributes; acquisition and deletion of system privileges;**

-actions taken by computer operators and system administrators and/or system security officers; **i.e., privileged operations such as the modification of TCB elements; accesses to TCB objects; changes of policy attributes of users, TCB configuration and security characteristics, and system privileges; selection and modification of audited events.**

**The events that are auditable by default, and those that are required for successful auditing of other events, which may not be disabled, shall be defined. The TCB shall provide a protected mechanism that displays the currently selected events and their defaults. The use of this mechanism shall be restricted to authorized system administrators.**

If availability policies are supported, attempts to circumvent or otherwise gain unauthorized access to resource-allocation limits shall be audited.

If non-discretionary access control policies are supported, the TCB shall be able to record any override of human-readable output markings. When the non-discretionary access control policies aim to control the flow of information between subjects, the TCB shall also be able to audit the identified event that may be used in the exploitation of covert channels.

3. 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 and policy attributes of the object (e.g., object security level).

4. **The TCB shall provide a protected mechanism to turn auditing on and off, and to select and change the events to be audited and their defaults, during the system operation. The use of this mechanism shall be restricted to authorized system administrators.** The system administrator shall be able to selectively audit the actions of one or more users based on individual identity and/or object policy attributes (e.g., object security level). **Audit review tools shall be available to authorized system administrators to assist in the inspection and review of audit data, and shall be protected from unauthorized use, modification, or destruction.**

**The TCB shall also provide protected audit-trail management functions that shall enable:**

**-creation, destruction, and emptying of audit trails; use of warning points regarding the size of the audit data, and modification of the audit trail size;**

**-formatting and compressing of event records;**

**-displaying of formatted audit trail data; and**

**-maintaining the consistency of the audit trail data after system failures and discontinuity of operation.**

## AD-3 Audit Tools

1. The TCB shall be able to create, maintain, and protect from modification or unauthorized access or destruction an audit trail of accesses to the objects it protects. The audit data shall be protected by the TCB so that read access to it is limited to those who are authorized for audit data.

2. The TCB shall be able to record the following types of events:

- use of the identification and authentication mechanisms, and system entry events;

- access control events selectable on a per user, per subject, per object, and/or per policy attribute basis; i.e., introduction of objects into a user's address space (e.g., file open, program initiation), creation and deletion of subjects and objects; distribution and revocation of access rights; changes of subject and object policy attributes; acquisition and deletion of system privileges;

-actions taken by computer operators and system administrators and/or system security officers; i.e., privileged operations such as the modification of TCB elements; accesses to TCB objects; changes of policy attributes of users, TCB configuration and security characteristics, and system privileges; selection and modification of audited events.

The events that are auditable by default, and those that are required for successful auditing of other events, which may not be disabled, shall be defined. The TCB shall provide a protected mechanism that displays the currently selected events and their defaults. The use of this mechanism shall be restricted to authorized system administrators.

If availability policies are supported, attempts to circumvent or otherwise gain unauthorized access to resource-allocation limits shall be audited.

If non-discretionary access control policies are supported, the TCB shall be able to record any override of human-readable output markings. When the non-discretionary access control policies aim to control the flow of information between subjects, the TCB shall also be able to audit the identified event that may be used in the exploitation of covert channels.

3. 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 and policy attributes of the object (e.g., object security level).

4. The TCB shall provide a protected mechanism to turn auditing on and off, and to select and change the events to be audited and their defaults, during the system operation. The use of this mechanism shall be restricted to authorized system administrators. The system administrator shall be able to selectively audit the actions of one or more users based on individual identity and/or object policy attributes (e.g., object security level). Audit review tools shall be available to authorized system administrators to assist in the inspection and review of audit data, and shall be protected from unauthorized use, modification, or destruction.

**The TCB shall provide tools for audit data processing. These shall include specifically designed tools: for verifying the consistency of the audit data; for verifying the selection of audit events; for audit trail management. The audit trail management tools shall enable:**

-creation, destruction, and emptying of audit trails; use of warning points regarding the size of the audit data, and modification of the audit trail size;

-formatting and compressing of event records;

-displaying of formatted audit trail data; and

-maintaining the consistency of the audit trail data after system failures and discontinuity of operation.

**5. Audit review tools shall be available to authorized users to assist in the inspection and review of audit data, and shall be protected from unauthorized modification or destruction. The TCB shall also provide tools for post-collection audit analysis (e.g., intrusion detection) that shall be able to selectively review (1) the actions of one or more users (e.g., identification, authentication, system-entry, and access control actions); (2) the actions performed on a specific object or system resource; and (3) all, or a specified set of, audited exceptions; and (4) actions associated with a specific policy attribute.The review tools shall be able to operate concurrently with the system operation.**

## AD-4 Audit Alarms

1. The TCB shall be able to create, maintain, and protect from modification or unauthorized access or destruction an audit trail of accesses to the objects it protects. The audit data shall be protected by the TCB so that read access to it is limited to those who are authorized for audit data.

2. The TCB shall be able to record the following types of events:

- use of the identification and authentication mechanisms, and system entry events;

- access control events selectable on a per user, per subject, per object, and/or per policy attribute basis; i.e., introduction of objects into a user’s address space (e.g., file open, program initiation), creation and deletion of subjects and objects; distribution and revocation of access rights; changes of subject and object policy attributes; acquisition and deletion of system privileges;

-actions taken by computer operators and system administrators and/or system security officers; i.e., privileged operations such as the modification of TCB elements; accesses to TCB objects; changes of policy attributes of users, TCB configuration and security characteristics, and system privileges; selection and modification of audited events.

The events that are auditable by default, and those that are required for successful auditing of other events, which may not be disabled, shall be defined. The TCB shall provide a protected mechanism that displays the currently selected events and their defaults. The use of this mechanism shall be restricted to authorized system administrators.

If availability policies are supported, attempts to circumvent or otherwise gain unauthorized access to resource-allocation limits shall be audited.

If non-discretionary access control policies are supported, the TCB shall be able to record any override of human-readable output markings. When the non-discretionary access control policies aim to control the flow of information between subjects, the TCB shall also be able to audit the identified event that may be used in the exploitation of covert channels.

**The TCB shall contain a mechanism that is able to monitor the occurrence or accumulation of auditable events that may indicate an imminent violation of the product's security policy. This mechanism shall be able to immediately notify the security administrator when thresholds are exceeded, and, if the occurrence or accumulation of these security relevant events continues, the system shall take the least disruptive action to terminate the event. That is, the TCB shall be able to send a message to the system console and/or the administrator’s terminal when thresholds are exceeded, or when audit records are unable to be recorded, and, if the occurrence or accumulation of these security-relevant events continue, the TCB shall generate an alarm (this shall be the default) or initiate a secure system shutdown.**

3. 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 and policy attributes of the object (e.g., object security level).

4. The TCB shall provide a protected mechanism to turn auditing on and off, and to select and change the events to be audited and their defaults, during the system operation. The use of this mechanism shall be restricted to authorized system administrators. The system administrator shall be able to selectively audit the actions of one or more users based on individual identity and/or object policy attributes (e.g., object security level). Audit review tools shall be available to authorized system administrators to assist in the inspection and review of audit data, and shall be protected from unauthorized use, modification, or destruction.

The TCB shall provide tools for audit data processing. These shall include specifically designed tools: for verifying the consistency of the audit data; for verifying the selection of audit events; for audit trail management. The audit trail management tools shall enable:

-creation, destruction, and emptying of audit trails; use of warning points regarding the size of the audit data, and modification of the audit trail size;

-formatting and compressing of event records;

-displaying of formatted audit trail data; and

-maintaining the consistency of the audit trail data after system failures and discontinuity of operation.

5. Audit review tools shall be available to authorized users to assist in the inspection and review of audit data, and shall be protected from unauthorized modification or destruction. The TCB shall also provide tools for post-collection audit analysis (e.g., intrusion detection) that shall be able to selectively review (1) the actions of one or more users (e.g., identification, authentication, system-entry, and access control actions); (2) the actions performed on a specific object or system resource; and (3) all, or a specified set of, audited exceptions; and (4) actions associated with a specific policy attribute.The review tools shall be able to operate concurrently with the system operation.

### AD-5 Real-Time Intrusion Detection

1. The TCB shall be able to create, maintain, and protect from modification or unauthorized access or destruction an audit trail of accesses to the objects it protects. The audit data shall be protected by the TCB so that read access to it is limited to those who are authorized for audit data.

2. The TCB shall be able to record the following types of events:

- use of the identification and authentication mechanisms, and system entry events;

- access control events selectable on a per user, per subject, per object, and/or per policy attribute basis; i.e., introduction of objects into a user's address space (e.g., file open, program initiation), creation and deletion of subjects and objects; distribution and revocation of access rights; changes of subject and object policy attributes; acquisition and deletion of system privileges;

-actions taken by computer operators and system administrators and/or system security officers; i.e., privileged operations such as the modification of TCB elements; accesses to TCB objects; changes of policy attributes of users, TCB configuration and security characteristics, and system privileges; selection and modification of audited events.

The events that are auditable by default, and those that are required for successful auditing of other events, which may not be disabled, shall be defined. The TCB shall provide a protected mechanism that displays the currently selected events and their defaults. The use of this mechanism shall be restricted to authorized system administrators.

If availability policies are supported, attempts to circumvent or otherwise gain unauthorized access to resource-allocation limits shall be audited.

If non-discretionary access control policies are supported, the TCB shall be able to record any override of human-readable output markings. When the non-discretionary access control policies aim to control the flow of information between subjects, the TCB shall also be able to audit the identified event that may be used in the exploitation of covert channels.

The TCB shall contain a mechanism that is able to monitor the occurrence or accumulation of auditable events that may indicate an imminent violation of the product's security policy. This mechanism shall be able to immediately notify the security administrator when thresholds are exceeded, and, if the occurrence or accumulation of these security relevant events continues, the system shall take the least disruptive action to terminate the event. That is, the TCB shall be able to send a message to the system console and/or the administrator's terminal when thresholds are exceeded, or when audit records are unable to be recorded, and, if the occurrence or accumulation of these security-relevant events continue, the TCB shall generate an alarm (this shall be the default) or initiate a secure system shutdown.

3. 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 and policy attributes of the object (e.g., object security level).

4. The TCB shall provide a protected mechanism to turn auditing on and off, and to select and change the events to be audited and their defaults, during the system operation. The use of this mechanism shall be restricted to authorized system administrators. The system administrator shall be able to selectively audit the actions of one or more users based on individual identity and/or object policy attributes (e.g., object security level). Audit review tools shall be available to authorized system administrators to assist in the inspection and review of audit data, and shall be protected from unauthorized use, modification, or destruction.

The TCB shall provide tools for audit data processing. These shall include specifically designed tools: for verifying the consistency of the audit data; for verifying the selection of audit events; for audit trail management. The audit trail management tools shall enable:

-creation, destruction, and emptying of audit trails; use of warning points regarding the size of the audit data, and modification of the audit trail size;

-formatting and compressing of event records;

-displaying of formatted audit trail data; and

-maintaining the consistency of the audit trail data after system failures and discontinuity of operation.

5. Audit review tools shall be available to authorized users to assist in the inspection and review of audit data, and shall be protected from unauthorized modification or destruction. The TCB shall also provide tools for post-collection audit analysis (e.g., intrusion detection) that shall be able to selectively review (1) the actions of one or more users (e.g., identification, authentication, system-entry, and access control actions); (2)

the actions performed on a specific object or system resource; and (3) all, or a specified set of, audited exceptions; and (4) actions associated with a specific policy attribute.The review tools shall be able to operate concurrently with the system operation.

**The TCB shall be able to perform real-time event reporting and intrusion detection in support of the product's security policy. The TCB shall include a real-time mechanism that is able to monitor the occurrence or accumulation of security-relevant events that may indicate an imminent security violation. This mechanism shall be able to generate an alarm when thresholds are exceeded and, if the occurrence or accumulation of these events persists, the TCB shall take the least disruptive action to terminate the event(s).**

### 4.3.5 Rated Access Control Components

Functional components implementing discretionary policies can be rated based on their scope (e.g., whether it includes all subjects and objects in a system, or only a defined subset; whether access control includes subject and object attributes), and on their coverage (e.g., their ability to control the propagation and retention of access rights for subjects and objects and their ability to encapsulate objects within a subject such that access to the object is allowed only by invoking the encapsulating subject.) In addition, discretionary policy rating can also refer to the ability to control access at a given subject granularity (e.g., at the individual user and group, or role level) and object granularity (e.g., memory partition, memory segment, file, record).

Non-discretionary access controls can be rated using the same generic levels as those used for discretionary policies. However, the granularity of subject and object to which non-discretionary access controls apply can be significantly finer than that of discretionary policies. Since non-discretionary policies control information flow, they must control access to object status attributes such as object size, existence, locking mode, and subject status attributes such as process-suspended or process-active indicators.

Separation-of-role policies can use existing access control functions of an IT product to implement its required rules. For this reason, separation-of-role policies can be rated using the same generic levels of subject and object granularity and scope as those used for discretionary and non-discretionary policies (discussed below and in Appendix C). In their simplest form, access control components implementing separation of role policies are rated by the separation of unprivileged subjects from those with administrative responsibilities. These component requirements include separation of product resources, of data, and of administrator-controlled policy attributes. The rating will take into account the granularity of separation between unprivileged subjects and those with administrative responsibilities.

In rating the access control components, four levels are identified using the definition of policies in Appendix C. The component rating reflected by these levels is based on the scope, granularity, and coverage of access control requirements. The choice of requirements at each level is largely guided by the access control characteristics of current commercially available products and by the goal of retaining the ability to harmonize these requirements with other existing standards.

Level AC-1 represents a minimal level of policy definition and enforcement. That is, the authorization rules apply to a defined subset of subjects and objects, and the administration of policy (i.e., access control) attributes cover only a subset of the functions defined at higher levels. Level AC-2 extends the coverage of access control policies and associated attributes of level AC-1 by recognizing that multiple policies could be supported within the same product. This level also extends the coverage of attribute administration largely to reflect object import and export. Level AC-3 enhances the scope of access control to all subjects and objects. Instead of referring to only a defined subset of subjects and objects, the requirements of this level refer to all subjects and objects. If non-discretionary policies that aim at controlling information flow are supported, then the requirement granularity at this level is extended to include all subject and object policy and status attributes. This level of access control is appropriate when non-discretionary policies are used that support information flow control. In such environments, lack of access control to subject and object status variables constitute a significant source of covert channels. However, this level retains the ability to define authorization and attribute administration on a per type-of-object basis. Level AC-4 extends the requirement coverage to include time- and location-based access controls, as well as inclusion and exclusion of user access rights whenever groups or roles are used. This level also extends the requirements for object and subject creation and destruction, adding explicit authorization, inheritance, space availability, and attribute inheritance conditions. It is expected that this level of access control would be used in products where fine-grain access control policies are required.

## AC-1 Minimal Access Control

1. **Definition of Access Control Attributes**

**The TCB shall define and protect access control attributes for subjects and objects. Subject attributes shall include named individuals or defined groups or both. Object attributes shall include defined access rights (e.g., read, write, execute) that can be assigned to subject attributes.**

2. **Administration of Access Control Attributes.**

**The TCB shall define and enforce rules for assignment and modification of access control attributes for subjects and objects. The effect of these rules shall be that access permission to an object by users not already possessing access permission is assigned only by authorized users. These rules shall allow authorized users to specify and control sharing of objects by named individuals or defined groups of individuals, or by both, and shall provide controls to limit propagation of access rights. These controls shall be capable of including or excluding access to the granularity of a single user.**

**If different rules of assignment and modification of access control attributes apply to different subjects and/or objects, the totality of these rules shall be shown to support the defined policy.**

3. **Authorization of Subject References to Objects**

**The TCB shall define and enforce authorization rules for the mediation of subject references to objects. These rules shall be based on the access control attributes of subjects and objects. These rules shall, either by explicit user action or by default, provide that objects are protected from unauthorized access.**

**The scope of the authorization rules shall include a defined subset of the product's subjects and objects and associated access control attributes. The coverage of authorization rules shall specify the types of objects and subjects to which these rules apply. If different rules apply to different subjects and objects, the totality of these rules shall be shown to support the defined policy.**

4. **Subject and Object Creation and Destruction**

**The TCB shall control the creation and destruction of subjects and objects. These controls shall include object reuse. That is, all authorizations to the information contained within a storage object shall be revoked prior to initial assignment, allocation or reallocation to a subject from the TCB's pool of unused storage objects; information, including encrypted representations of information, produced by a prior subjects' actions shall be unavailable to any subject that obtains access to an object that has been released back to the system.**

5. **Object Encapsulation**

**If the TCB supports mechanisms for object encapsulation, controls must be available for: (1) access authorization to encapsulated objects; (2) creation of encapsulated subsystems by users; and (3) invocation of encapsulated subsystems.**

## AC-2 Basic Access Control

### 1. Definition of Access Control Attributes

The TCB shall define and protect access control attributes for subjects and objects. Subject attributes shall include named individuals or defined groups or both. Object attributes shall include defined access rights (e.g., read, write, execute) that can be assigned to subject attributes. **If multiple access control policies are supported, the access control attributes corresponding to each individual policy shall be identified.**

**The subject and object attributes shall accurately reflect the sensitivity and/or integrity of the subject or object.**

### 2. Administration of Access Control Attributes

The TCB shall define and enforce rules for assignment and modification of access control attributes for subjects and objects. The effect of these rules shall be that access permission to an object by users not already possessing access permission is assigned only by authorized users. These rules shall allow authorized users to specify and control sharing of objects by named individuals or defined groups of individuals, or by both, and shall provide controls to limit propagation of access rights. These controls shall be capable of including or excluding access to the granularity of a single user.

**The rules for assignment and modification of access control attributes shall include those for attribute assignment to objects during import and export operations (e.g., import of non-labeled sensitive data, export of labeled information).** If different rules of assignment and modification of access control attributes apply to different subjects and/or objects, the totality of these rules shall be shown to support the defined policy.

### 3. Authorization of Subject References to Objects

The TCB shall define and enforce authorization rules for the mediation of subject references to objects. These rules shall be based on the access control attributes of subjects and objects. These rules shall, either by explicit user action or by default, provide that objects are protected from unauthorized access.

The scope of the authorization rules shall include a defined subset of the product's subjects and objects and associated access control attributes. The coverage of authorization rules shall specify the types of objects and subjects to which these rules apply. If different rules apply to different subjects and objects, the totality of these rules shall be shown to support the defined policy.

**If multiple policies are supported, the authorization rules for each policy shall be defined separately. The TCB shall define and enforce the composition of policies, including the enforcement of the authorization rules (e.g., subject and object type coverage, enforcement precedence).**

**4. Subject and Object Creation and Destruction**

The TCB shall control the creation and destruction of subjects and objects. These controls shall include object reuse. That is, all authorizations to the information contained within a storage object shall be revoked prior to initial assignment, allocation or reallocation to a subject from the TCB's pool of unused storage objects; information, including encrypted representations of information, produced by a prior subjects' actions shall be unavailable to any subject that obtains access to an object that has been released back to the system.

**5. Object Encapsulation**

If the TCB supports mechanisms for object encapsulation, controls must be available for: (1) access authorization to encapsulated objects; (2) creation of encapsulated subsystems by users; and (3) invocation of encapsulated subsystems

## AC-3 Extended Access Control

**1. Definition of Access Control Attributes**

The TCB shall define and protect access control attributes for subjects and objects. Subject attributes shall include named individuals or defined groups or both. Object attributes shall include defined access rights (e.g., read, write, execute) that can be assigned to subject attributes. If multiple access control policies are supported, the access control attributes corresponding to each individual policy shall be identified.

The subject and object attributes shall accurately reflect the sensitivity and/or integrity of the subject or object. **The TCB shall immediately notify a terminal user of each attribute change of any subject associated with that user during an interactive session that reflects a change in the sensitivity or integrity of that session (e.g., a change of the user's security level). A terminal user shall be able to query the TCB as desired for a display of the subject's complete set of access control attributes (e.g., the complete sensitivity label).**

**The TCB shall support the assignment of access control attributes (e.g., minimum and maximum security levels) to all attached physical devices. These attributes shall be used by the TCB to enforce constraints imposed by the physical environments in which the devices are located.**

### 2. Administration of Access Control Attributes

The TCB shall define and enforce rules for assignment and modification of access control attributes for subjects and objects. The effect of these rules shall be that access permission to an object by users not already possessing access permission is assigned only by authorized users. These rules shall allow authorized users to specify and control sharing of objects by named individuals or defined groups of individuals, or by both, and shall provide controls to limit propagation of access rights. These controls shall be capable of including or excluding access to the granularity of a single user.

The rules for assignment and modification of access control attributes shall include those for attribute assignment to objects during import and export operations (e.g., import of non-labeled sensitive data, export of labeled information). If different rules of assignment and modification of access control attributes apply to different subjects and/or objects, the totality of these rules shall be shown to support the defined policy.

### 3. Authorization of Subject References to Objects

The TCB shall define and enforce authorization rules for the mediation of subject references to objects. These rules shall be based on the access control attributes of subjects and objects. These rules shall, either by explicit user action or by default, provide that objects are protected from unauthorized access.

The scope of the authorization rules shall include **all** subjects, **storage** objects **(e.g., processes, segments, devices)** and associated access control attributes **that are directly or indirectly accessible to subjects external to the TCB. If non-discretionary access control policies are used that aim to control the flow of information between subjects, the scope of the authorization rules shall also include all policy and status attributes of subjects and storage objects (e.g., quotas, object existence, size, access time, creation and modification time, locked/unlocked).** If different rules apply to different subjects and objects, the totality of these rules shall be shown to support the defined policy.

If multiple policies are supported, the authorization rules for each policy shall be defined separately. The TCB shall define and enforce the composition of policies, including the enforcement of the authorization rules (e.g., subject and object type coverage, enforcement precedence).

### 4. Subject and Object Creation and Destruction

The TCB shall control the creation and destruction of subjects and objects. These controls shall include object reuse. That is, all authorizations to the information contained within a storage object shall be revoked prior to initial assignment, allocation, reallocation to a subject from the TCB's pool of unused storage objects; information, including encrypted representations of information, produced by a prior subjects' actions shall be unavailable to any subject that obtains access to an object that has been released back to the system.

### 5. Object Encapsulation

If the TCB supports mechanisms for object encapsulation, controls must be available for: (1) access authorization to encapsulated objects; (2) creation of encapsulated subsystems by users; and (3) invocation of encapsulated subsystems.

## AC-4 Fine-Grain Access Control

### 1. Definition of Access Control Attributes

The TCB shall define and protect access control attributes for subjects and objects. Subject attributes shall include named individuals or defined groups or both. Object attributes shall include defined access rights (e.g., read, write, execute) that can be assigned to subject attributes. If multiple access control policies are supported, the access control attributes corresponding to each individual policy shall be identified. **The subject's access control attributes also shall include time and location attributes that can be assigned to authenticated user identities.**

The subject and object attributes shall accurately reflect the sensitivity and/or integrity of the subject or object. The TCB shall immediately notify a terminal user of each attribute change of any subject associated with that user during an interactive session that reflects a change in the sensitivity or integrity of that session (e.g., a change of the user's security level). A terminal user shall be able to query the TCB as desired for a display of the subject's complete set of access control attributes (e.g., the complete sensitivity label).

The TCB shall support the assignment of access control attributes (e.g., device labels) to all attached physical devices. These attributes shall be used by the TCB to enforce constraints imposed by the physical environments in which the devices are located.

### 2. Administration of Access Control Attributes

The TCB shall define and enforce rules for assignment and modification of access control attributes for subjects and objects. The effect of these rules shall be that access permission to an object by users not already possessing access permission is assigned only by authorized users. These rules shall allow authorized users to specify and control sharing of objects by named individuals or defined groups of individuals, or by both, and shall provide controls to limit propagation of access rights **(i.e., these rules shall define the distribution, revocation, and review of access control attributes)**. **The controls defined by these rules shall be capable of specifying for each named object, a list of individuals and a list of groups of named individuals, with their respective access rights to that object. Furthermore, for each named object, it shall be possible to specify a list of named individuals and a list of groups of named individuals for which no access to the object is given. These controls shall also be capable of specifying access-time dependency (i.e., the effect of the**

**distribution and revocation of access control attributes take place at a certain time and shall last for a specified period of time), and/or access-location dependency (i.e., shall specify the locations from which the distribution and revocation of privileges shall take place).**

The rules for assignment and modification of access control attributes shall include those for attribute assignment to objects during import and export operations (e.g., import of non-labeled sensitive data, export of labeled information). If different rules of assignment and modification of access control attributes apply to different subjects and/or objects, the totality of these rules shall be shown to support the defined policy.

### 3. Authorization of Subject References to Objects

The TCB shall define and enforce authorization rules for the mediation of subject references to objects. These rules shall be based on the access control attributes of subjects and objects. These rules shall, either by explicit user action or by default, provide that objects are protected from unauthorized access. **These rules shall include time-of-access and location-of-access controls defined for subjects and objects.**

The scope of the authorization rules shall include all subjects, storage objects (e.g., processes, segments, devices) and associated access control attributes that are directly or indirectly accessible to subjects external to the TCB. If non-discretionary access control policies are used that aim to control the flow of information between subjects, the scope of the authorization rules shall also include all policy and status attributes of subjects and storage objects (e.g., quotas, object existence, size, access time, creation and modification time, locked/unlocked). If different rules apply to different subjects and objects, the totality of these rules shall be shown to support the defined policy.

If multiple policies are supported, the authorization rules for each policy shall be defined separately. The TCB shall define and enforce the composition of policies, including the enforcement of the authorization rules (e.g., subject and object type coverage, enforcement precedence).

### 4. Subject and Object Creation and Destruction

The TCB shall **define and enforce rules for** the creation and destruction of subjects and objects. **The controls defined by these rules shall be capable of specifying for each subject and object: (1) creation and destruction authorization; (2)** object reuse; **(3) space availability (i.e., storage space shall be available for the creation of a subject and object); (4) default subject or object attributes and attribute inheritance rules (if any).**

**The rules for subject and object creation and destruction shall specify their coverage in terms of the types of objects and subjects to which they apply. If different rules and conditions apply to different subjects and objects, the totality of these rules shall be shown to support the defined policy properties. If multiple policies are supported, these rules shall define the composition of policies and how the conditions of the subject and object creation and destruction are enforced (e.g., subject and object type coverage, enforcement precedence).**

**5. Object Encapsulation**

If the TCB supports mechanisms for object encapsulation, controls must be available for: (1) access authorization to encapsulated objects; (2) creation of encapsulated subsystems by users; and (3) invocation of encapsulated subsystems.

### 4.3.5.1 Rated Covert Channel Handling Components

Covert channel handling requires that functions must be added to the software and/or hardware and firmware elements of a TCB to help deter the use of, limit the bandwidth of, or eliminate, covert channels. The rating of the covert channel handling components is based both on the scope of these requirements and their coverage (e.g., elimination, bandwidth limitation, audit, administrative control, applicability to timing channels or storage channels). The scope of level CCH-1 is limited to storage channels and the coverage is limited to functions that deter covert channel use. Coverage is extended at level CCH-2 by the addition of requirements of bandwidth limitation and storage channel elimination for common system configurations. Level CCH-3 extends the requirements of level CCH-2 by including all channels, not just covert storage channels.

#### CCH-1 Deterrence of Storage Channel Use

**1. The TCB and privileged applications shall include functions that help audit the use of covert storage channels. These functions shall enable the identification of the transmitter, receiver, and specific covert channels used (e.g., TCB and privileged application element used to transmit information).**

**2. The functions added to the TCB and privileged applications for storage channel auditing shall be identified for each channel and shall be available in common product configurations. If audit functions are not added to certain storage channels (e.g., hardware storage channels), evidence must be provided to justify why these channels do not represent a security threat for the intended use of the product.**

#### CCH-2 Storage Channel Audit and Bandwidth Limitation

1. The TCB and privileged applications shall include functions that help audit the use of covert storage channels. These functions shall enable the identification of the transmitter, receiver, and specific covert channels used (e.g., TCB and privileged application element used to transmit information). **TCB functions that help limit the bandwidth and/or eliminate covert storage channels shall also be provided. The bandwidth limits for each channel shall be settable by system administrators.**

2. The functions added to the TCB and privileged applications for storage channel auditing shall be identified for each channel and shall be available in common product configurations. If audit functions are not added to certain storage channels (e.g., hardware storage channels), evidence must be provided to justify why these channels do not represent a security threat for the intended use of the product. **TCB and privileged application functions that help limit the bandwidth and/or eliminate covert storage channels shall also be available in common product configurations.**

**If channel bandwidth limitation and channel elimination functions are not added to certain storage channels (e.g., hardware storage channels), evidence must be provided to justify why these channels do not represent a security threat for the intended use of the product.**

**CCH-3 Timing Channel Audit and Bandwidth Limitation**

1. The TCB and privileged applications shall include functions that help audit the use of covert storage channels. These functions shall enable the identification of the transmitter, receiver, and specific covert channels used (e.g., TCB and privileged application element used to transmit information). TCB functions that help limit the bandwidth and/or eliminate covert storage channels shall also be provided. The bandwidth limits for each channel shall be settable by system administrators.

2. The functions added to the TCB and privileged applications for storage channel auditing shall be identified for each channel and shall be available in common product configurations. If audit functions are not added to certain storage channels (e.g., hardware storage channels), evidence must be provided to justify why these channels do not represent a security threat for the intended use of the product. TCB and privileged application functions that help limit the bandwidth and/or eliminate covert storage **or timing** channels shall also be available in common product configurations.

If channel bandwidth limitation and channel elimination functions are not added to certain storage **or timing** channels (e.g., hardware channels), evidence must be provided to justify why these channels do not represent a security threat for the intended use of the product.

### 4.3.6 Rated Resource Allocation Components

The resource allocation component rating is concerned with the extent and strength of containment control exerted over the availability and distribution of product resources. The resource allocation components are rated based on the scope of containment (e.g., defined set of resources versus all resources) and the coverage of containment (e.g., resource restrictions, control, priorities, audit).

Level AR-1 defines basic requirements of resource allocation restrictions in terms of a specified subset of system resources, subjects and objects. Level AR-2 extends the scope of resource control to all system resources and increases the coverage of the resource allocation features by requiring the auditing and signaling of attempted violations of resource allocation limits (or quotas). Level AR-3 further extends the coverage of the resource allocation features by introducing the requirement for prioritized allocation.

**AR-1 Resource Restrictions**

> **The TCB shall provide the capability to place restrictions on the number of subjects and objects a user may have allocated at any given time. The TCB shall control a defined set of system resources (e.g., memory, disk space) such that no one individual user can deny access to another user's subject and object space. All subjects, objects, and resources shall be defined with default space or time quotas and quantity-of-resources attributes.**

**AR-2 Complete Resource Control**

> The TCB shall provide the capability to place restrictions on the number of subjects and objects a user may have allocated at any given time. The TCB shall control a defined set of system resources (e.g., memory, disk space) such that no one individual user can deny access to another user's subject and object space. All subjects, objects, and resources shall be defined with default space or time quota and number-of-resources attributes. **An individual user shall be unable to deny access to any system resource by means of circumventing resource-allocation limits, or otherwise manipulating the TCB, so as to restrict the TCB's ability to offer services to other users and objects.**

**AR-3 Prioritized Resource Allocations**

> The TCB shall provide the capability to place restrictions on the number of subjects and objects a user may have allocated at any given time. The TCB shall control a defined set of system resources (e.g., memory, disk space) such that no one individual user can deny access to another user's subject and object space. All subjects, objects, and resources shall be defined with default space or time quotas and quantity-of-resources attributes. An individual user shall be unable to deny access to any system resource by means of circumventing resource-allocation limits, or otherwise manipulating the TCB, so as to restrict the TCB's ability to offer services to other users and objects. **The TCB shall include resource-allocation priorities among the subject attributes. Each subject shall be granted a priority against which the TCB shall allocate resources. The TCB shall**

**mediate resource-allocation priorities in such a manner that access requirements of the TCB and high-priority subjects shall be fulfilled first, in a prioritized manner. All resources within the TCB (hardware and software) shall be controlled in pre-assigned blocks.**

### 4.3.7 Rated Security Management Components

The rating of the security-management components is based primarily on the coverage, and strength of these components. For example, level SM-3 is considered to be stronger than level SM-2 because the separation of administrative and operator roles offers added resistance to accidents or misdeeds. Level SM-3 also extends the coverage of level SM-2 because it reflects the use of a wider policy coverage. Level SM-4 extends the coverage and strength of level SM-3 because (1) it requires the availability of trusted tools for security management (e.g., tools offering a graphical interface to the administrator, tools enhancing system administration, and tools enabling the administrator to perform consistency checking), and (2) it further limits through fine-grain separation of administrative roles the potential damage that can be caused by error or misdeed.

#### SM-1 Minimal Security Management

**1. The TCB shall provide an installation mechanism for the setting and updating of its configuration parameters, and for the initialization of its protection-relevant data structures before any user or administrator policy attributes are defined. It shall allow the configuration of TCB internal databases and tables.**

**2. The TCB shall provide protected mechanisms for displaying and modifying the security policy parameters.**

**3. The TCB shall provide protected mechanisms for manually displaying, modifying, or deleting user registration and account parameters. These parameters shall include unique user identifiers, their account, and their associated user name and affiliation. The TCB shall allow the manual enabling and disabling of user identities and/or accounts.**

**4. The TCB shall provide protected mechanisms for routine control and maintenance of system resources. That is, it shall allow the enabling and disabling of peripheral devices, mounting of removable storage media, backing-up and recovering user objects; maintaining the TCB hardware and software elements (e.g., on site testing); and starting and shutting down the system.**

**5. The use of the protected mechanisms for system administration shall be limited to authorized administrative users.**

**SM-2 Basic Security Management**

1. The TCB shall provide an installation mechanism for the setting and updating of its configuration parameters, and for the initialization of its protection-relevant data structures before any user or administrator policy attributes are defined. It shall allow the configuration of TCB internal databases and tables.

**The TCB shall distinguish between normal mode of operation and maintenance mode, and shall provide a maintenance-mode mechanism for recovery and system start-up.**

2. The TCB shall provide protected mechanisms for displaying and modifying the security policy parameters. **These parameters shall include identification, authentication, system entry and access control parameters for the entire system and for individual users.**

**The TCB shall have a capability to define the identification and authentication policy on a system-wide basis (e.g., password minimum and maximum lifetime, password length and complexity parameters). The TCB mechanisms shall have the capability to limit: (1) maximum period of interactive session inactivity, (2) maximum login or session time, and (3) successive unsuccessful attempts to log in to the system.**

**If availability policies are supported, the TCB shall provide a mechanism to control the availability of system resources via resource quotas and quantity-of-resources limits.**

3. The TCB shall provide protected mechanisms for manually displaying, modifying, or deleting user registration and account parameters. These parameters shall include unique user identifiers, their account, and their associated user name and affiliation. The TCB shall allow the manual enabling and disabling of user identities and/or accounts.

**The TCB shall provide a means to uniquely identify security policy attributes. It shall also provide a means of listing all these attributes for a user, and all the users associated with an attribute. It shall be capable of defining and maintaining the security policy attributes for subjects including: defining and maintaining privileges for privileged subjects, discretionary and non-discretionary attributes (e.g., definition and maintenance of group, role, and secrecy and/or integrity level membership), and centralized distribution, review and revocation of policy attributes.**

4. The TCB shall provide protected mechanisms for routine control and maintenance of system resources.It shall allow the enabling and disabling of peripheral devices, mounting of removable storage media, backing-up and recovering user objects; maintaining the TCB hardware and software elements (e.g., on site testing); and starting and shutting down the system.

5. The use of the protected mechanisms for system administration shall be limited to authorized administrative users.

## SM-3 Policy-oriented Security Management

1. The TCB shall provide an installation mechanism for the setting and updating of its configuration parameters, and for the initialization of its protection-relevant data structures before any user or administrator policy attributes are defined. It shall allow the configuration of TCB internal databases and tables.

The TCB shall distinguish between normal mode of operation and maintenance mode, and shall provide a maintenance-mode mechanism for recovery and system start-up. **This mechanism shall include a means to initialize administrative privileges and administrative identification, authentication, and system-entry attributes.**

2. The TCB shall provide protected mechanisms for displaying and modifying the security policy parameters. These parameters shall include identification, authentication, system entry and access control parameters for the entire system and for individual users.

The TCB shall have a capability to define the identification and authentication policy on a system-wide basis (e.g., password minimum and maximum lifetime, password length and complexity parameters). The TCB mechanisms shall have the capability to limit: (1) maximum period of interactive session inactivity, (2) maximum login or session time, and (3) successive unsuccessful attempts to log in to the system. **The TCB shall provide an administrative capability to specify the authentication method on a per policy-attribute basis whenever multiple identification and authentication methods are used; e.g., via user passwords, tokens, or biometrics.**

**If the TCB is designed to support multiple login sessions per user identity, the administrators shall be able to limit the number of simultaneous login sessions on an authorization-attribute basis.**

**The TCB shall also have a capability to limit the successive unsuccessful attempts to login from a specific port of entry, and/or with a specific user identity or account.**

If availability policies are supported, the TCB shall provide a mechanism to control the availability of system resources via resource quotas and quantity-of-resources limits.

3. The TCB shall provide protected mechanisms for manually displaying, modifying, or deleting user registration and account parameters. These parameters shall include unique user identifiers, their account, and their associated user name and affiliation. **The TCB shall allow the automatic disabling of user identities and/ or accounts, after a period during which the identity and/or account have not been used. The time period shall be administrator specified, with a specified default provided. The TCB shall allow the automatic re-enabling of disabled user identities and/or accounts after an administrator-specified period of time.**

The TCB shall provide a means to uniquely identify security policy attributes. It shall also provide a means of listing all these attributes for a user, and all the users associated with an attribute. It shall be capable of defining and maintaining the security policy attributes for subjects including: defining and maintaining privileges for privileged subjects, discretionary and non-discretionary attributes (e.g., definition and maintenance of group, role, and secrecy and/or integrity level membership), and centralized distribution, review and revocation of policy attributes.

4. **The TCB shall support separate operator and administrator functions. The operator functions shall be restricted to those necessary for performing routine operations. The operator functions** shall allow the enabling and disabling of peripheral devices, mounting of removable storage media, backing-up and recovering user objects; maintaining the TCB hardware and software elements (e.g., on-site testing); and starting and shutting down the system.

5. The use of the protected mechanisms for system administration shall be limited to authorized administrative users.

## SM-4 Extended Security Management

1. The TCB shall provide an installation mechanism for the setting and updating of its configuration parameters, and for the initialization of its protection-relevant data structures before any user or administrator policy attributes are defined. It shall allow the configuration of TCB internal databases and tables.

The TCB shall distinguish between normal mode of operation and maintenance mode, and shall provide a maintenance-mode mechanism for recovery and system start-up. This mechanism shall include a means to initialize administrative privileges and administrative identification, authentication, and system-entry attributes.

2. The TCB shall provide protected mechanisms for displaying and modifying the security policy parameters. These parameters shall include identification, authentication, system entry and access control parameters for the entire system and for individual users.

The TCB shall have a capability to define the identification and authentication policy on a system-wide basis (e.g., password minimum and maximum lifetime, password length and complexity parameters). The TCB mechanisms shall have the capability to limit: (1) maximum period of interactive session inactivity, (2) maximum login or session time, and (3) successive unsuccessful attempts to log in to the system. The TCB shall provide an administrative capability to specify the authentication method on a per policy-attribute basis whenever multiple identification and authentication methods are used; e.g., via user passwords, tokens, or biometrics.

If the TCB is designed to support multiple login sessions per user identity, the administrators shall be able to limit the number of simultaneous login sessions on an authorization-attribute basis.

The TCB shall also have a capability to limit the successive unsuccessful attempts to login from a specific port of entry, and/or with a specific user identity or account.

If availability policies are supported, the TCB shall provide a mechanism to control the availability of system resources via resource quotas and quantity-of-resources limits.

3. The TCB shall provide protected mechanisms for manually displaying, modifying, or deleting user registration and account parameters. These parameters shall include unique user identifiers, their account, and their associated user name and affiliation. The TCB shall allow the automatic disabling of user identities and/or accounts, after a period during which the identity and/or account have not been used. The time period shall be administrator specified, with a specified default provided. The TCB shall allow the automatic re-enabling of disabled user identities and/or accounts after an administrator-specified period of time.

The TCB shall provide a means to uniquely identify security policy attributes. It shall also provide a means of listing all these attributes for a user, and all the users associated with an attribute. It shall be capable of defining and maintaining the security policy attributes for subjects including: defining and maintaining privileges for privileged subjects, discretionary and non-discretionary attributes (e.g., definition and maintenance of group, role, and secrecy and/or integrity level membership), and centralized distribution, review and revocation of policy attributes.

**The TCB shall provide trusted tools for system administration. These shall include: tools for verifying the consistency of the user registration and system configuration; tools for verifying the proper system installation; tools for verifying that the TCB does not contain extraneous programs and data.**

**The TCB shall include tools for determining whether the TCB is in a secure initial state after start-up and recovery.**

**The TCB shall include tools for verifying the consistency of users, subject, and objects policy attributes (e.g., cross checks between subject and object attributes and registered user attributes).**

4. The TCB shall support separate operator and administrator functions. The operator functions shall be restricted to those necessary for performing routine operations. The operator functions shall allow the enabling and disabling of peripheral devices, mounting of removable storage media, backing-up and recovering user objects; maintaining the TCB hardware and software elements (e.g., on-site testing); and starting and shutting down the system. **The administrative functions shall support separate security administrator and auditor roles. The TCB shall enable the administrators to perform their functions only after taking a distinct**

**auditable action to assume an administrator role. Non-security functions that can be performed in the security administrative role shall be limited strictly to those essential to performing the security role effectively.**

5. The use of the protected mechanisms **and tools** for system administration shall be limited to authorized administrative users.

### 4.3.8 Rated Reference Mediation Components

The rating of the reference mediation components are largely based on scope and granularity of references. At level RM-1, the scope of mediation is limited to a defined subject and object subset (i.e., the same subset as that defined by the access control components). At level RM-2, the scope of mediation is extended to the complete set of subjects and objects. At level RM-3, the granularity of references includes defined subsets, or all: (1) objects, (2) object policy attributes (e.g., access rights, security levels, quotas); and (3) object status attributes (e.g., object existence, length, locking state). Level RM-4 is derived by requiring a model of privilege mediation. This level extends the coverage of level RM-3 and is intended for use in a TCB that can be extended with privileged processes of various applications.

#### RM-1 Mediation of References to a Defined Subject/Object Subset

**1. The TCB shall mediate all references to subjects, objects, resources, and services (e.g., TCB functions) described in the TCB specifications. The mediation shall ensure that all references are directed to the appropriate security-policy functions.**

**2. Reference mediation shall include references to the defined subset of subjects, objects, and resources protected under the TCB security policy, and to their policy attributes (e.g., access rights, security and/or integrity levels, role identifiers).**

**3. References issued by privileged subjects shall be mediated in accordance with the policy attributes defined for those subjects.**

#### RM-2 Mediation of References to all Subjects and Objects

1. The TCB shall mediate all references to subjects, objects, resources, and services (e.g., TCB functions) described in the TCB specifications. The mediation shall ensure that all references are directed to the appropriate security-policy functions.

2. Reference mediation shall include **control of** references to **all** subjects, objects, and resources protected under the TCB security policy, and to their policy attributes (e.g., access rights, security and/or integrity levels, role identifiers, quotas).

3. References issued by privileged subjects shall be mediated in accordance with the policy attributes defined for those subjects.

**RM-3 Mediation of References to Subject and Object Attributes**

1. The TCB shall mediate all references to subjects, objects, resources, and services (e.g., TCB functions) described in the TCB specifications. The mediation shall ensure that all references are directed to the appropriate security-policy functions.

2. Reference mediation shall include control of references to all subjects, objects, and resources protected under the TCB security policy, **to their policy** (e.g., access rights, security and/or integrity levels, role identifiers, quotas) **and status** attributes **(e.g., existence, length, locking state).**

3. References issued by privileged subjects shall be mediated in accordance with the policy attributes defined for those subjects.

**RM-4 Mediation of Privileged Subject References**

1. The TCB shall mediate all references to subjects, objects, resources, and services (e.g., TCB functions) described in the TCB specifications. The mediation shall ensure that all references are directed to the appropriate security-policy functions.

2. Reference mediation shall include control of references to all subjects, objects, and resources protected under the TCB security policy, to their policy (e.g., access rights, security and/or integrity levels, role identifiers, quotas) and status attributes (e.g., existence, length, locking state).

3. References issued by privileged subjects shall be mediated in accordance with **the privilege model** defined for those subjects.

### 4.3.9 Rated Logical TCB Protection Components

The rating of the TCB protection components is based on the coverage of TCB requirements. Level P-1 of TCB protection has two basic requirements, namely TCB isolation and noncircumventability of TCB isolation functions. Level P-2 extends the coverage of level P-1 with the requirements of ensuring the consistency of TCB global variables and the elimination of undesirable TCB dependencies on unprivileged user actions. These additional requirements help eliminate large classes of TCB penetration means. Level P-3 eliminates an additional class of penetration means that is generally more difficult to exploit than those classes addressed in the previous two levels. The intent of these levels is to reflect increasingly better functions for TCB penetration resistance.

### P-1 Basic TCB Isolation

**The TCB shall maintain a domain for its own execution that protects it from external interference and tampering (e.g., by reading or modification of its code and data structures). The protection of the TCB shall provide TCB isolation and noncircumventability of TCB isolation functions as follows:**

**1. TCB Isolation requires that (1) the address spaces of the TCB and those of unprivileged subjects are separated such that users, or unprivileged subjects operating on their behalf, cannot read or modify TCB data structures or code, (2) the transfers between TCB and non-TCB domains are controlled such that arbitrary entry to or return from the TCB are not possible; and (3) the user or application parameters passed to the TCB by addresses are validated with respect to the TCB address space, and those passed by value are validated with respect to the values expected by the TCB.**

**2. Noncircumventability of TCB isolation functions requires that the permission to objects (and/or to non-TCB data) passed as parameters to the TCB are validated with respect to the permissions required by the TCB, and references to TCB objects implementing TCB isolation functions are mediated by the TCB.**

### P-2 TCB Isolation and Consistency

The TCB shall maintain a domain for its own execution that protects it from external interference and tampering (e.g., by reading or modification of its code and data structures). The protection of the TCB shall provide TCB isolation and noncircumventability of TCB isolation functions as follows:

1. TCB Isolation requires that (1) the address spaces of the TCB and those of unprivileged subjects are separated such that users, or unprivileged subjects operating on their behalf, cannot read or modify TCB data structures or code, (2) the transfers between TCB and non-TCB domains are controlled such that arbitrary entry to or return from the TCB are not possible; and (3) the user or application parameters passed to the TCB by addresses are validated with respect to the TCB address space, and those passed by value are validated with respect to the values expected by the TCB.

2. Non-circumventability of TCB isolation functions requires that the permission to objects (and/or to non-TCB data) passed as parameters to the TCB are validated with respect to the permissions required by the TCB, and references to TCB objects implementing TCB isolation functions are mediated by the TCB.

**TCB protection shall also maintain the consistency of TCB global variables and eliminate undesirable dependencies of the TCB on unprivileged subject or user actions.**

**3. Consistency of TCB global variables requires that consistency conditions defined over TCB internal variables, objects, and functions hold before and after any TCB invocation.**

**4. Elimination of undesirable dependencies of the TCB on unprivileged subject actions requires that any TCB invocation by an unprivileged subject (or user) input to a TCB call may not place the TCB in a state such that it is unable to respond to communication initiated by other users.**

## P-3 TCB Isolation and Timing Consistency

The TCB shall maintain a domain for its own execution that protects it from external interference and tampering (e.g., by reading or modification of its code and data structures). The protection of the TCB shall provide TCB isolation and noncircumventability of TCB isolation functions as follows:

1. TCB Isolation requires that (1) the address spaces of the TCB and those of unprivileged subjects are separated such that users, or unprivileged subjects operating on their behalf, cannot read or modify TCB data structures or code, (2) the transfers between TCB and non-TCB domains are controlled such that arbitrary entry to or return from the TCB are not possible; and (3) the user or application parameters passed to the TCB by addresses are validated with respect to the TCB address space, and those passed by value are validated with respect to the values expected by the TCB.

2. Non-circumventability of TCB isolation functions requires that the permission to objects (and/or to non-TCB data) passed as parameters to the TCB are validated with respect to the permissions required by the TCB, and references to TCB objects implementing TCB isolation functions are mediated by the TCB.

TCB protection shall also maintain the consistency of TCB global variables and eliminate undesirable dependencies of the TCB on unprivileged subject or user actions.

3. Consistency of TCB global variables requires that consistency conditions defined over TCB internal variables, objects, and functions hold before and after any TCB invocation.

4. Elimination of undesirable dependencies of the TCB on unprivileged subject actions requires that any TCB invocation by an unprivileged subject (or user) input to a TCB call may not place the TCB in a state such that it is unable to respond to communication initiated by other users.

**Furthermore, TCB protection shall maintain the timing consistency of condition checks.**

**5. Timing consistency of condition checks requires that a validation check holds at the instant when the TCB action depending on that check is performed.**

### 4.3.10 Rated Physical TCB Protection Components

The rating of the physical TCB protection is determined by the coverage and strength of the physical protection requirements; i.e., on the ability to prevent, deter, detect, and counter physical attacks against the product. Level PP-1 requires the availability of physical protection functions and devices to support administrative and environment measures of controlling access to the TCB of the product. Level PP-2 extends the coverage of this requirement by specifying that employed functions and devices shall have the ability to unambiguously detect any attempt of physical tampering regardless of its outcome. Level PP-3 increases the strength of the physical TCB protection by requiring the use of physical countermeasures with well-defined work factors. The intent of these requirements is to distinguish between physical protection supporting administrative measures, tamper-detection functions, and tamper-resistance functions.

#### PP-1 Administrative and Environment Protection

**1. Administrative procedures and environmental features necessary for establishing the physical security of a product's TCB shall be defined.**

**2. Product functions and devices necessary to establish physical control over the product's TCB shall be identified and provided.**

#### PP-2 Detection of Physical Attack

1. Administrative procedures and environmental features necessary for establishing the physical security of a product's TCB shall be defined.

2. Product functions and devices necessary to establish physical control over the product's TCB shall be identified and provided. **TCB devices allowing the unambiguous detection of physical tampering shall be employed. These devices shall be shown to be physically tamper-resistant and noncircumventable.**

#### PP-3 Physical and Environmental Countermeasures

1. Administrative procedures and environmental features necessary for establishing the physical security of a product's TCB shall be defined.

2. Product functions and devices necessary to establish physical control over the product's TCB shall be identified and provided. **TCB devices that provide countermeasures to physical tampering shall be employed. The strength of these devices shall be determined based on well-defined work factor parameters relevant to the supported policies. For confidentiality policies, these devices shall resist disclosure via theft, inspection of physical media, wiretapping, and/or analysis of product emanations. For integrity policies, these devices shall resist modification of hardware functionality and modification of stored data via mechanical methods and/or electronic jamming. For availability policies, these devices shall resist loss of service via anticipated environmental stress (e.g., water damage, fire, vibration, impact) or other forms of physical attack.**

### 4.3.11 Rated TCB Self Checking Components

The TCB self-checking components are rated based on the scope of the checking performed (i.e., hardware and/or firmware versus software) and on the coverage of the checking methods (i.e., periodic or continuous checking). At level SC-1, a minimal level of self-checking is required (e.g., similar to those currently available on most commercial workstations). Level SC-2 extends these requirements by including power-on self tests, loadable tests, and operator-controlled tests that are used to periodically validate the correct operation of the TCB hardware and/or firmware elements. The scope of these tests is extended at level SC-3 by the addition of configurable software and/or firmware functions that perform periodic self tests. At level SC-4, the self-test coverage is extended by requiring that hardware, firmware, and/or software self tests be performed continuously during the product operation.

#### SC-1 Minimal Self Checking

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

#### SC-2 Basic Self Checking

Hardware and/or software features shall be provided that can be used to periodically validate the correct operation of the on-site hardware and firmware elements of the TCB. **These features shall include: power-on tests, loadable tests, and operator-controlled tests.**

**The power-on tests shall test all basic components of the TCB hardware and firmware elements including memory boards and memory interconnections; data paths; busses; control logic and processor registers; disk adapters; communication ports; system consoles, and the keyboard speaker. These tests shall cover all components that are necessary to run the loadable tests and the operator-controlled tests.**

**The loadable tests shall cover: processor components (e.g., arithmetic and logic unit, floating point unit, instruction decode buffers, interrupt controllers, register transfer bus, address translation buffer, cache, and processor-to-memory bus controller); backplane busses; memory controllers; and writable control memory for operator-controlled and remote system-integrity testing.**

**Operator-controlled tests shall be able to initiate a series of one-time or repeated tests, to log the results of these tests and, if any fault is detected, to direct the integrity-test programs to identify and isolate the failure.**

## SC-3 Software-Test Support

Hardware and/or software features shall be provided that can be used to periodically validate the correct operation of the on-site hardware and firmware elements of the TCB. These features shall include: power-on tests, loadable tests, and operator-controlled tests.

The power-on tests shall test all basic components of the TCB hardware and firmware elements including memory boards and memory interconnections; data paths; busses; control logic and processor registers; disk adapters; communication ports; system consoles, and the keyboard speaker. These tests shall cover all components that are necessary to run the loadable tests and the operator-controlled tests.

The loadable tests shall cover: processor components (e.g., arithmetic and logic unit, floating point unit, instruction decode buffers, interrupt controllers, register transfer bus, address translation buffer, cache, and processor-to-memory bus controller); backplane busses; memory controllers; and writable control memory for operator-controlled and remote system-integrity testing.

Operator-controlled tests shall be able to initiate a series of one-time or repeated tests, to log the results of these tests and, if any fault is detected, to direct the integrity-test programs to identify and isolate the failure.

**Configurable software or firmware features shall be provided that can be used to validate the correct operation of the on-site software elements (i.e., code and data structures) of the TCB. These features may include, but are not limited to, checksums and consistency checks for TCB elements stored on storage media (e.g., disk-block consistency conditions).**

## SC-4 Continuous Software-Test Support

Hardware and/or software features shall be provided that can be used to periodically validate the correct operation of the on-site hardware and firmware elements of the TCB. These features shall include: power-on tests, loadable tests, and operator-controlled tests.

The power-on tests shall test all basic components of the TCB hardware and firmware elements including memory boards and memory interconnections; data paths; busses; control logic and processor registers; disk adapters; communication ports; system consoles, and the keyboard speaker. These tests shall cover all components that are necessary to run the loadable tests and the operator-controlled tests.

The loadable tests shall cover: processor components (e.g., arithmetic and logic unit, floating point unit, instruction decode buffers, interrupt controllers, register transfer bus, address translation buffer, cache, and processor-to-memory bus controller); backplane busses; memory controllers; and writable control memory for operator-controlled and remote system-integrity testing.

Operator-controlled tests shall be able to initiate a series of one-time or repeated tests, to log the results of these tests and, if any fault is detected, to direct the integrity-test programs to identify and isolate the failure.

Configurable software or firmware features shall be provided that can be used to validate the correct operation of the on-site software elements (i.e., code and data structures) of the TCB. These features may include, but are not limited to, checksums and consistency checks for TCB elements stored on storage media (e.g., disk-block consistency conditions).

**Tests that detect possible inconsistencies of the TCB elements (i.e., data structures and code) shall be performed whenever the content or structure of these elements are modified as consequence of a transient failure during an unprivileged subject's action.**

### 4.3.12 Rated TCB Start-Up and Recovery Components

The TCB start-up and recovery components are rated based on feature coverage; i.e., whether manual (levels TR-1, TR-2) or automatic (level TR-3) recovery and start-up in a secure state is provided, and whether the loss of user objects during recovery can be minimized (level TR-5) or just detected (level TR-4). Primitive forms of secure recovery, where potentially all objects are lost during recovery, have a narrower coverage than that intended to be provided by automated procedures.

### TR-1 Minimal Requirements for Recovery or Start-up

**1. Procedures and/or mechanisms shall be provided to assure that, after a TCB failure or other discontinuity, recovery without protection compromise is obtained.**

### TR-2 Basic Requirements for Recovery or Start-up

1. Procedures and/or mechanisms shall be provided to assure that, after a TCB failure or other discontinuity, recovery without protection compromise is obtained.

**2. If automated recovery and start-up is not possible, the TCB shall enter a state where the only system access method is via administrative interfaces, terminals, or procedures. Administrative procedures shall exist to restore the system to a secure state (i.e., a state in which all the security-policy properties hold).**

### TR-3 Automated Recovery or Start-up

1. Procedures and/or mechanisms shall be provided to assure that, after a TCB failure or other discontinuity, recovery without protection compromise is obtained.

2. **Automated procedures, under the control of the TCB, shall be provided to assure that after a system failure, other discontinuity, or start-up, a secure state is obtained without undue loss of system or user objects. The security policy properties, or requirements, used to determine that a secure state is obtained shall be defined.**

### TR-4 Object-Loss Detection

1. Procedures and/or mechanisms shall be provided to assure that, after a TCB failure or other discontinuity, recovery without protection compromise is obtained.

2. Automated procedures, under the control of the TCB, shall be provided to assure that after a system failure, other discontinuity, or start-up, a secure state is obtained without undue loss of system or user objects. The security policy properties, or requirements, used to determine that a secure state is obtained shall be defined. **The TCB shall include checkpoint functions for recovery. Upon recovery, it shall be possible to discover which user objects are corrupted or unaccessible due to the TCB failure, if any, and to automatically notify the users.**

**TR-5 Object-Loss Minimization**

1. Procedures and/or mechanisms shall be provided to assure that, after a TCB failure or other discontinuity, recovery without protection compromise is obtained.

2. Automated procedures, under the control of the TCB, shall be provided to assure that after a system failure, other discontinuity, or start-up, a secure state is obtained without undue loss of system or user objects. The security policy properties, or requirements, used to determine that a secure state is obtained shall be defined. The TCB shall include checkpoint functions for recovery. Upon recovery, it shall be possible to discover which user objects are corrupted or unaccessible due to the TCB failure, if any, and to automatically notify the users. **The TCB functions that can be invoked through the TCB interface shall be atomic (i.e., shall have the property that either their invocation is completed correctly or the recovered system state should be the one immediately prior to the execution of the TCB function). The recovered secure state should minimize the corruption and inaccessibility of user objects due to the TCB failure.**

### 4.3.13 Rated TCB Privileged Operation Components

The TCB privileged operation components are rated based on the granularity of privilege associated with individual TCB functions or groups of functions (level PO-1), with modules of TCB functions and operations of administrative roles (level PO-2), with individual actions (level PO-3), and with individual code sections of an action (level PO-4). The intent of these ratings is to separate (1) fine granularity of privileges from coarser granularity and (2) the static association of privileges with functions and modules from the run-time association of privileges with actions (i.e., function invocations) and sections of code within actions. Although the granularity of privileges of a product is a design choice, the intent of these requirements is to encourage use of fine granularity of privilege and run-time association of privileges, at least for the TCB actions of bypassing access controls.

**PO-1 Privilege Association with TCB Functions**

**1. TCB privileges needed by individual functions, or groups of functions, shall be identified. Privileged TCB calls or access to privileged TCB objects, such as user and group registration files, password files, security and integrity-level definition file, role definition file, or audit-log file shall also be identified.**

**2. The identified privileged functions of a TCB functional component shall be associated only with the privileges necessary to complete their task.**

### PO-2 Privilege Association with TCB Modules

1. TCB privileges needed by individual functions, or groups of functions, of a functional component shall be identified. Privileged TCB calls or access to privileged TCB objects, such as user and group registration files, password files, security and integrity-level definition file, role definition file, audit-log file shall also be identified. **It shall be possible to associate TCB privileges with TCB operations performed by administrative users.**

**2.The modules of a TCB function shall be associated only with the privileges necessary to complete their task.**

**3. Support for product privilege implementation and association with TCB modules provided by lower-level mechanisms or procedures (e.g., operating system, processors, language) shall be provided.**

### PO-3 Privilege Association with Individual Actions

1. TCB privileges needed by individual functions, or groups of functions, of a functional component shall be identified. Privileged TCB calls or access to privileged TCB objects, such as user and group registration files, password files, security and integrity-level definition file, role definition file, audit-log file shall also be identified. It shall be possible to associate TCB privileges with TCB operations performed by administrative users.

2**.** The modules of a TCB function shall be associated only with the privileges necessary to complete their task.**TCB privileges needed by individual actions of a module (i.e., function invocations) shall be identified (e.g., privileges shall be assigned to actions that bypass access controls, such as disclosure and modification of user objects). Each action shall be associated only with the privileges necessary to complete its task.**

3. Support for product privilege implementation and association with TCB **actions** provided by lower-level mechanisms or procedures (e.g., operating system, processors, language) shall be provided.

### PO-4 Dynamic Privilege Association with Individual Actions

1. TCB privileges needed by individual functions, or groups of functions, of a functional component shall be identified. Privileged TCB calls or access to privileged TCB objects, such as user and group registration files, password files, security and integrity-level definition file, role definition file, audit-log file shall also be identified. It shall be possible to associate TCB privileges with TCB operations performed by administrative users.

2. TCB privileges needed by actions of a functional component (i.e., function invocations) shall be identified (e.g., privileges shall be assigned to actions that bypass access controls, such as disclosure and modification of user objects). Each action shall be associated only with the privileges necessary to complete its task. **The identified TCB privileges shall be**

**used by each functional component to restrict the propagation of errors and failures of security mechanisms that may lead to protection policy violations. TCB functions allowing each component to acquire individual privileges up to the maximum necessary and allowed, and to drop those privileges (e.g., functions implementing privilege bracketing) shall be defined. These functions shall be used to limit the use of privileges that allow the bypassing of security policy controls within the TCB.**

**3.** Support for product privilege implementation and association with TCB actions provided by lower-level mechanisms or procedures (e.g., operating system, processors, language) shall be provided**.**

### 4.3.14 Rated TCB Ease-of-Use Components

The rating of the TCB ease of use components reflects the scope and coverage of the protection functions in covering common product configurations. At level EU-1, the requirements reflect the general need for special administrative functions, not merely using an editor to modify administrative files or default options for security parameters. The coverage of the ease-of-use requirements is extended at level EU-2 by providing for fail-safe defaults and user-settable defaults for defined (privileged and unprivileged) subjects and objects, and the means by which applications can protect themselves and their objects from unauthorized use. The scope of the requirements is extended at levels EU-3 and EU-4 by enlarging the set of subjects and objects affected by this requirement to include subjects and objects of common configurations, and all subjects and objects, respectively.

#### EU-1 Ease of Security Management

**1. The TCB shall provide well-defined actions to undertake administrative functions. Default options shall be provided for security parameters of administrative functions.**

#### EU-2 Ease of Application Programming

1. The TCB shall provide well-defined actions to undertake administrative functions. Default options shall be provided for security parameters of administrative functions.

**The TCB shall include fail-safe defaults for the policy attributes of the defined subjects and objects, as well as user-settable defaults for the defined subjects and objects.**

**2. The TCB shall provide well-defined application programming interfaces and programming functions (e.g., libraries) for all its policies to support the development of applications that can define and enforce security policies on application-controlled subjects and objects. The TCB shall enable user-controlled reduction of access rights available to applications.**

### EU-3 Common Configuration Coverage

1. The TCB shall provide well-defined actions to undertake administrative functions. **Fail-safe** default options shall be provided for security parameters of administrative functions.

**The TCB shall include fail-safe defaults for the policy attributes of subjects, objects (e.g., devices) and services used in common system configurations, as well as user-settable defaults for these subjects and objects.**

2. The TCB shall provide well-defined application programming interfaces and programming functions (e.g., libraries) for all its policies to support the development of applications that can define and enforce security policies on application-controlled subjects and objects. The TCB shall enable user-controlled reduction of access rights available to applications.

### EU-4 Complete Configuration Coverage

1. The TCB shall provide well-defined actions to undertake administrative functions. Fail-safe default options shall be provided for security parameters of administrative functions.

**The TCB shall include fail-safe, user-settable defaults for the policy attributes of all subjects, objects (e.g., devices), and services.**

2. The TCB shall provide well-defined application programming interfaces and programming functions (e.g., libraries) for all its policies to support the development of applications that can define and enforce security policies on application-controlled subjects and objects. The TCB shall enable user-controlled reduction of permissions available to applications.

## 4.4 Bibliographic Notes

TBD.

# Chapter 5.

# DEVELOPMENT ASSURANCE REQUIREMENTS

## 5.1 Overview

Development assurance is concerned with showing that a specific IT product satisfies the functional requirements of a protection profile. This chapter defines assurance requirements that are used in protection profiles to define an IT product developer's (i.e., producer's) responsibilities in establishing the correctness of the product's security functions. These requirements are partitioned into components that identify unique concerns a developer must address during the product design, implementation, documentation, support, and maintenance. By addressing these concerns, the developer can increase consumer and evaluator confidence that the product satisfies the functional requirements of a protection profile. Varying degrees of confidence can be established using different combinations and subsets of the assurance components.

The assurance components defined in this standard have evolved from computer security and engineering experience in demonstrating the correctness of IT hardware and software protection functions. The components also include requirements of existing criteria and reflect the interpretations of those requirements in practice during the past decade. The components are specified in a product-independent manner and, thus, are applicable to a wide set of products and protection functions. To enable the profile developers to establish varying degrees of confidence in the correctness of product protection functions, each assurance component is rated based on a set of well-defined parameters. These ratings can also help establish the relationships between, and the harmonization of, the assurance requirements defined by this standard and those of existing standards.

This chapter is divided into four sections. The remainder of this section defines four *classes* of development assurance components and describes the types of components in each class. The second section presents a description of each type of assurance component. The third section contains the rated assurance components. The last section includes a bibliography of useful literature references. (Appendices D and E present some of the technical underpinnings used in deriving the requirements of the modular decomposition and penetration analysis components.)

**Classes of Development Assurance.** The development assurance components have been partitioned into four classes reflecting distinct product development tasks: (1) development process, (2) operational support, (3) development environment, and (4) development evidence. The four classes, and associated components, are illustrated in Figure 5.

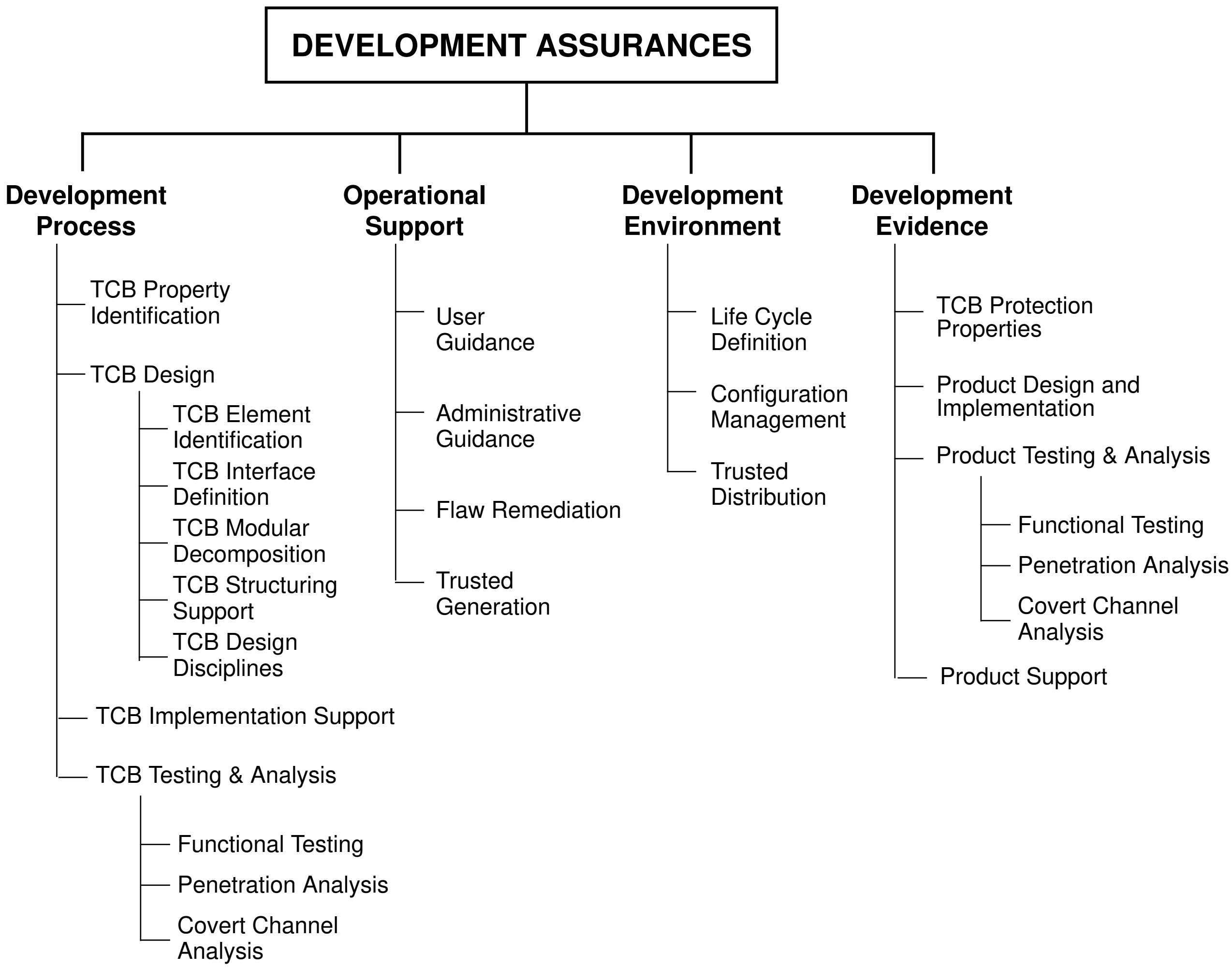



**Figure 5. Taxonomy of Development Assurances.**

**Development Process.** The development process class consists of the following four assurance components that identify specific activities the developer must undertake during the design, implementation, and analysis of an IT product: (1) TCB property identification, (2) TCB design, which includes TCB element identification, interface definition, modular decomposition, structuring support, and design disciplines used, (3) TCB implementation support, and (4) TCB testing and analysis, which includes security functional testing, penetration analysis, and covert channel analysis. Since the development process includes the primary assurances for the correct implementation of protection functions, its components are included in most profiles. The selection of different component levels within a profile is determined by the assurance goals established for the profile, by the dependencies among assurance components, and by the dependencies between the profile functional and assurance components.

**Operational Support.** The operational support class consists of assurance components that a developer must satisfy to enable users to operate the product securely. This class includes the following four components: (1) user guidance, (2) administrative guidance, (3) flaw remediation, and (4) trusted generation. These components require the developer to convey clearly the operational procedures for TCB generation, installation, operation, and flaw correction. They also require the developers to provide tools and/or procedures to properly install and configure the product. The first two components of this class are included in all profiles whereas the last two are included in profiles that target medium and high-assurance products. The selection of different component levels within a profile is largely determined by assurance goals and by the dependencies among assurance components.

**Development Environment.** The development environment class consists of assurance components that refer to quality of the development, maintenance, and distribution-control process for secure products. This class includes the following three components: (1) life cycle definition, (2) configuration management, and (3) trusted distribution of the product. These components require that the developer enforces a discernible engineering process to develop and maintain a product, establishes control over the product configuration during development and maintenance, and employs technical measures for the detection or prevention of uncontrolled TCB modification during product distribution. A key assurance aspect of these components is the extent to which the development process and operational support requirements are integrated into the developer's engineering processes. The development environment components are included in profiles that target medium and high-assurance products. The selection of different component levels within a profile is largely determined by assurance goals and by the dependencies among assurance components.

**Development Evidence.** The development evidence class consists of assurance components that describe the documentation that a developer must produce and maintain to show that the other assurance requirements have been satisfied. The components of the development evidence class establish the level of detail and scope of the developer documentation and include requirements for evidence of: (1) TCB protection properties, (2) product design and implementation, (3) product testing and analysis, and (4) product support. These components are included in all profiles. The selection of different component levels within a profile is determined exclusively by the dependencies among assurance components, since these levels must mirror to a large extent the levels of the other development assurance components.

## 5.2 Development Assurance Components

### 5.2.1 Development Process

#### 5.2.1.1 TCB Property Identification

The identification of TCB properties is the *prima facie* assurance that the consistency of the TCB's behavior with respect to the protection profile's functional requirements can be established. These properties are the baseline set of protection claims for a TCB. They enable the generation of test conditions for security policy analysis and penetration testing. They also help define the IT product's protection capabilities in product documentation.

The first step to demonstrate that a product satisfies the functional requirements of a protection profile is to produce the description of the TCB protection properties. This is achieved by (1) identification of the TCB elements intended to implement the functional requirements, and (2) justification of how and why the identified elements implement these requirements. Repeating this step for each functional requirement of a protection profile produces a description of the set of protection properties. Since a functional requirement can be satisfied by different product architectures and operating systems, the set of protection properties will illustrate both the developer's philosophy of protection and the protection architecture choices made.

Demonstrating the consistency of the TCB behavior with the requirements of the profile functional components can be performed with different degrees of rigor. For example, consistency verification can be performed by an informal process of tracing the requirements within a product's TCB and providing a simple description of the claimed TCB property. This informal process relies primarily on informal functional requirements (i.e., as provided by the protection profile) and on descriptions of TCB elements. The primary assurance gained from this informal process is derived from the consistency and coherence of the profile requirements, and from the explanation of why the TCB elements satisfy those requirements. This explanation will reveal whether the developer's interpretation of the profile functional requirements in the product TCB is valid.

The degree of rigor with which the demonstration of consistency between the TCB elements and the functional requirements can be performed increases whenever formal or informal models of the functional requirements are used. Models capture the essence of the functional requirements by providing policy properties that must be maintained by the TCB. For example, the Bell-LaPadula Model contains two policy properties of mandatory access control. These examples are the **-property* (star property), which allows a subject write access to an object only if the security level of the subject equals that of the object, and the *simple security condition*, which allows a subject read access to an object only if the security level of the subject dominates that of the object. A discretionary access control model may have a policy property stating that "only the owner of an object can distribute or review permissions to that object." A TCB isolation model may have an isolation property stating that "a parameter passed by address to a TCB function invocation may only refer to the invoker's space, and not to the TCB space or to another subject space."

Tracing precise requirement properties among the TCB elements is likely to be significantly more effective than tracing profile requirement descriptions, which are informally expressed. However, the precision with which the requirement properties are expressed by a model generally depends on the type of model and the degree of model formalism. Furthermore, the tracing process itself also depends on the degree of precision and formalism with which the TCB elements are described. For example, precision is enhanced by providing Descriptive Interface Specifications (DIS) instead of informal descriptions of the TCB interface found in product reference manuals, or by providing Formal Interface Specifications (FIS) instead of the DIS. The use of formal models and DIS/FIS of the TCB makes it possible to perform the tracing process by (in)formally interpreting the requirements model in the DIS/FIS. By showing that a documented (in)formal interpretation of a requirement model in a TCB is valid, the properties of the TCB can be stated (in)formally with a higher degree of rigor.

### 5.2.1.2 TCB Design

The TCB design component comprises several subcomponents that provide product development assurance. These subcomponents are (1) element identification, (2) interface definition, (3) modular decomposition, (4) structuring support, and (5) design disciplines. Details of each component follow.

#### 5.2.1.2.1 TCB Element Identification

The importance of the TCB element identification as an assurance component derives from the fact that all other assurance methods rely on it either directly or indirectly. Intuitively, the TCB includes all code and data structures that implement protection functions of a TCB (i.e., functional components). Although this intuitive definition of the TCB elements is precise, in practice the identification of these elements can be a challenging activity for the following three reasons.

First, TCBs may include code and data structures that are irrelevant to the protection components. In practice, products often implement functions and mechanisms that include security irrelevant elements and whose main purpose is not protection. Separating the protection relevant from the irrelevant elements within these functions can sometimes be a very difficult task because of the complexity and performance implications of the interfaces that are introduced between the protection relevant and irrelevant elements of a function. Although desirable for assurance purposes, in general, it may be impractical to remove all security irrelevant code from the TCB.

Second, the TCB is defined with respect to a set of protection requirements and a set of assurances necessary to demonstrate that the TCB satisfies those requirements. A product may include protection functions for which assurance is not required. Nevertheless, these protection functions in practice become part of the TCB since they affect the overall behavior of the product in other environments. For example, separating different TCBs for functions implementing different security policies may be impractical. The TCB of a product may include protection functions to support confidentiality, integrity, and availability to various degrees, but the only required policy support assurances may be for confidentiality. Providing a separate TCB for availability and integrity components is impractical for most products and unjustifiable, based on the incremental assurance benefits that might be derived from the removing these components from the confidentiality TCB. In practice, the determination of which components must be included in a TCB cannot be made exclusively based on the specific-policy relevance of the code and data structures.

Third, sometimes it is challenging to determine whether a functional component is security-policy relevant without the benefit of a formal model of a security policy. For example, if a formal state-transition model is available, any TCB function whose execution may cause a state transition is, by definition, security-relevant. However, in the absence of a formal model, one can determine whether a function is security-policy relevant only if the function implements security policy checks. Among the functions that invoke other functions implementing security or accountability checks indirectly, few are required to be part of the TCB since, by definition, they could be placed outside the TCB and could invoke a TCB system call.

Since many of the IT product TCBs will include both protection-relevant and irrelevant elements, the identification of these elements (1) must separate the protection-relevant elements from the irrelevant ones (if any), and (2) must provide a rationale for the retention of the protection-irrelevant elements within the TCB.

#### 5.2.1.2.2 TCB Interface Definition

To analyze the protection of the TCB domain, one must first define interface of the TCB to external subjects. This interface establishes the boundary between the TCB elements and unprivileged subjects. The TCB protection behavior is defined at this interface.

The definition of the TCB interface is required by several assurance methods, including security and penetration analysis and testing, interpretation of security requirements and models within a TCB, and covert channel analysis and testing. Establishing the TCB interface requires that all TCB elements be identified to determine whether they present an interface (i.e., are visible) to an unprivileged subject.

The TCB interfaces typically consist of several components including (1) the command interfaces, (2) the application programming interfaces (i.e., system calls), and (3) the machine/processor interface (i.e., processor instructions). The command interface consists of the set of TCB commands that can be input via user-oriented devices such as keyboards, mouse devices, and joysticks; other command interfaces to a TCB may include various sensor input interfaces for real-time devices and processes external to the TCB. The application programming interface consists of all the TCB system calls that an application program can make, to include the signal, trap, and fault interfaces which an application program may invoke. Similarly, the machine/processor interface to the TCB consists of all instructions that refer to TCB internal data structures, (e.g., memory registers, segment and page tables), and processor registers (e.g., process status registers, segment, page table, capability, cache, and address-translation registers).

Determining the TCB interface cannot be simply performed by listing all commands, system calls, and processor instructions. Not all commands, system calls, or instructions may, in fact, represent a TCB interface. For example, some commands and library calls may refer to programs and data structures that are in user space. Similarly, some instructions may refer to operands that are already loaded into user-addressable registers and, therefore, need not include memory protection checks. Some command and application program interfaces may overlap and not represent distinct TCB interfaces. For example, two distinct command interfaces that are implemented by command processors running in user space may invoke the same application program interfaces of the TCB. Consequently, the two distinct commands do not provide distinct TCB interfaces. In some products, the TCB includes the entire application and presents a command interface to its users with no distinct application program interface or processor interface. For example, in some real-time, process-control systems, the external TCB interface may represent sensor input, but no external user or application program input. In this case the TCB components and the TCB external interface must still be identified because of attempts by external processes to provide the sensor input.

The TCB interface definition requires that all TCB functions which are visible outside the TCB be defined, including their calling conventions, parameters, parameter types, order, and exceptions signalled. The parameter types must include, in addition to the call parameters, all of the subjects, objects, and access control attributes affected by that call. Whenever covert channel analysis, penetration analysis, and resource-constraint analysis are required, the TCB interface definition must also include all effects of a call, including the direct visibility and alterability of internal TCB variables and functions. In these cases, the traditional definitions of TCB interfaces provided in product reference manuals must be augmented by additional elements. In all cases, all TCB interfaces must be included. No interface may remain undocumented, and no temporary interfaces for testing or performance monitoring, for example, should be included.

#### 5.2.1.2.3 TCB Modular Decomposition

Modular design and implementation constitutes a sound engineering practice in general, and therefore, this technique represents sound engineering practice for IT product development assurance. Reading, understanding, maintaining, testing, and evolving a software product is helped by modularity. "Understanding" includes identifying product parts and their relationships, and determining important product properties. Although modular design and implementation is not a security-specific assurance, products that employ it offer the following assurance advantages: (1) an incremental, divide-and-conquer approach to determining correctness properties; (2) an incremental, divide-and-conquer approach to product development, with many individuals per development team possible; (3) replacement independence of product parts based on well-defined interfaces and uniform reference (i.e., references to modules need not change when the modules change); and (4) an intuitive packaging of product components with ease of navigation through the product, module by module.

Appendix D provides some of the technical underpinnings used in deriving the requirements of the TCB modular decomposition.

#### 5.2.1.2.4 TCB Structuring Support

The TCB structuring using modular decomposition is necessary for understanding, maintaining, testing, and evolving a product. However, the modular decomposition does not necessarily reflect the run-time enforcement of TCB structuring since the separation of modules may not necessarily be supported by run-time mechanisms. The run-time enforcement of internal TCB structuring adds a measure of assurance that the TCB elements that are critical to the enforcement of the protection functions are separated from non-critical elements. Also, the use of run-time enforcement of TCB structuring helps separate protection-critical elements of the TCB from each other, thereby helping enforce the separation of protection concerns and minimizing the common mechanisms shared between protection critical elements.

Run-time enforcement of TCB structuring is useful in cases when compile-time structuring either cannot be enforced (e.g., the programming language does not enforce modular decomposition) or can be circumvented by transient hardware failures. In either case, software errors may propagate through the entire TCB and corrupt protection-critical elements. However, run-time enforcement of TCB structuring is not considered to be a protection function because it does not directly counter any security threat posed by unprivileged subjects. Unlike the support for the least-privilege TCB operation, which reduces the possibility that the penetration of a TCB functional component affects other components, the run-time support for TCB structuring has no direct protection use. Use of processor mechanisms to support TCB structuring is desirable for minimizing the performance penalties that will undoubtedly arise if these mechanisms were provided by run-time software.

A key assurance requirement for run-time mechanisms that support TCB structuring is that of *conceptual simplicity* and *well-defined semantics*. Conceptually simple mechanisms are generally easy to understand, and describe, define, or formally specify. Well-defined semantics enable the rigorous analysis of TCB structuring and increase the confidence that the internal TCB structuring is enforced correctly. The use of architecture features for run-time enforcement of TCB structuring into security kernels and privileged processes ranges from *process-isolation* features to the use of *ring* or *domain-of-protection* mechanisms. Among the architecture and operating system features employed for enforcing the TCB structuring, *segmentation* and *paging* has been used to separate logically distinct storage objects with separate access-control attributes. The separation of subsystem and module data structures and code within a TCB is sometimes supported by *ring* or *domain-of-protection* mechanisms with separate entry point support and protection (illustrated by ring or domain *gates*). Thus, the TCB can be described in terms of the different *rings* or *domains* of protection it employs for its structuring into subsystems and modules, and in terms of the segmentation or paging mechanisms it employs for structuring its internal code and data structures into logically distinct storage objects.

### 5.2.1.2.5 TCB Design Disciplines

Modularly decomposing the TCB provides many benefits. However, it does not minimize the complexity of the TCB or remove the protection-irrelevant elements from the TCB. Leaving protection-irrelevant elements within a TCB necessarily results in a significantly larger assurance effort because these elements must be included in the TCB’s analysis. Furthermore, modularly decomposing the TCB does not necessarily minimize the sharing of global variables between modules (i.e., the data structures used in a module need not be “hidden” within the module), and does not necessarily help layer the TCB (e.g., the cyclic dependencies among TCB modules cause lower layers to require services of higher layers). Consequently, the analysis of the TCB could become very complex for medium size (e.g., 500K to 1M lines of source code) or large products (e.g., over 1M lines of source code).

Several design disciplines enable the rigorous analysis of TCB security properties which is necessary for high-assurance products. First, the complexity of the TCB must be reduced by minimizing the number of protection-irrelevant elements that are left within the TCB. This requirement, together with the functional requirements of reference mediation and TCB protection, is the basis for demonstrating that the TCB implements the *reference validation mechanism*, which is important for the rigorous analysis of access control policy implementations (see Appendix B).

Second, the TCB structuring must employ the use of data hiding to minimize the sharing of global variables within the TCB. This requirement, together with the use of other design abstractions such as functional and control abstractions, significantly enhances the ability to structure the modules of a TCB into sets of (ordered) layers and to precisely determine the protection properties of those layers (e.g., derive abstraction and layer properties). As a result, the formal analysis of the TCB modules becomes possible.

Third, extensive use of high-level synchronization constructs, such as monitors and message passing, makes the analysis of the TCB behavior possible despite the occurrence of asynchronous events. Further structuring of TCB processes into threads decreases the cost of using processes as a TCB structuring mechanism, thereby enhancing the development of TCBs containing small independent modules sharing process space.

#### 5.2.1.3 Implementation Support

The implementation support component is an assurance method that can be used to simplify the task of establishing the correspondence between the product as built and the product design. The ultimate test of the development process applied to a TCB is how well the TCB implementation satisfies the protection profile requirements. Testing can establish that the TCB implementation exhibits at least the properties needed to satisfy the profile requirements. Analysis, however, is needed to establish that the implementation does no more than the profile requires. At a minimum, a complete analysis requires that the source code be available. For more detailed analysis, the system architecture and design are also necessary to simplify the task of tracking the required TCB properties from requirements down to implementation. As more rigor is brought to the process of design, more analysis can be done at higher levels of abstraction. To complete the chain and effectively leverage the previous analysis, the implementation should be organized and packaged in the same manner as the design to simplify the process of mapping the design to the implementation.

#### 5.2.1.4 TCB Testing and Analysis

A significant measure of product development assurance is derived from the methods used to test and analyze a TCB provides. These methods, which include (1) functional testing, (2) penetration analysis, and (3) covert channel analysis, are described below.

##### 5.2.1.4.1 Functional Testing

Functional testing is an assurance method for establishing that the TCB interface exhibits the properties necessary to satisfy the requirements of its protection profile. Functional testing is especially valuable in providing assurance that the TCB satisfies *at least* its functional protection requirements. It does establish that the TCB does *no more* than expected. The developer's functional testing objective is to uncover all design and implementation flaws that would enable a user external to the TCB to violate the product security policy. Such flaws would invalidate the developer's claim of compliance with the protection profile. The developer should perform functional testing whenever the TCB changes as a result of design analysis, independent evaluation, product evolution, or repair of security flaws identified by either consumers or previous functional testing.

All approaches to security functional testing require the following four major steps:

**Test plan development**. The test plans consist of test conditions, test data including the expected test outcomes, and test coverage analysis. Test plans must be developed for all TCB primitives exported at the TCB interface.

**Test program development**. The test programs developed must reflect the test conditions, test data, and coverage described in the test plans.

**Test procedure description**. The test procedures provide instructions for using individual test programs, and complete test suites.

**Test result analysis**. The analysis of the test results verifies that the test outputs correspond to the expected outcomes defined in the test plans.

Functional testing should be done on a copy of the TCB that is configured and installed as recommended in product documentation. The product should be operating in a normal mode, as opposed to maintenance or test mode. Tests should be done using user-level programs that cannot read or write internal TCB data structures or programs. New data structures and programs should also not be added to a TCB for security testing purposes, and special TCB entry points that are unavailable to external user programs should not be used. If a TCB is tested in maintenance mode using programs that cannot be run at the user level, the security tests would be meaningless because assurance cannot be gained that the TCB performs user-level access control correctly. If user-level test programs could read, write or add internal TCB data structures and programs, as would be required by traditional instrumentation testing techniques, the TCB would lose its isolation properties. If user-level test programs could use special TCB entry points not normally available to external users, the TCB would become circumventable in the normal mode of operation.

Functional testing should be conducted according to procedures defined in a test plan. Significant events during testing should be placed in a test log. As testing proceeds sequentially through each test case, the developer's test team should identify flaws and deficiencies that will need to be corrected. After changing TCB elements (i.e., hardware, firmware, or software) to correct these flaws and deficiencies, the developer should repeat the tests that identified problem(s) as well as any other tests related to the changed TCB elements. When the development team has corrected all functional problems and has analyzed and retested all corrections, a test report should be written and made a part of a functional testing report.

#### 5.2.1.4.2 Penetration Analysis

The penetration analysis of a computer product is a separate assurance concern from security policy design and/or implementation. Different TCBs may exhibit the same degree of penetration resistance, but implement widely different security policies, or may implement the same policies, but exhibit different degrees of penetration resistance. Furthermore, penetration analysis is an important assurance component since the effectiveness of all security policies rely on the penetration resistance of a TCB.

The penetration analysis of a TCB consists of the identification and confirmation of flaws in the design and implementation of protection functions that can be exploited by unprivileged users or application programs. Unlike security policy analysis and security functional testing, penetration analysis identifies TCB flaws that are not necessarily related to security policy design and implementation. For example, penetration analysis identifies vulnerabilities of reference mediation and TCB protection functions independent of the functions that implement security policy support. This implies that the type of policy that controls the subjects' access to objects is relevant to security functional analysis and testing, but not to penetration testing. Instead, penetration testing concerns include whether TCB elements may be surreptitiously viewed or modified, and whether TCB internal functions, which are intended to be invisible outside the TCB, can in fact be invoked under the control of unprivileged users or applications. Furthermore, penetration analysis includes assessments of the strength of TCB protection functions and of vulnerabilities of protection function implementation and operational use.

Appendix E presents some of the technical underpinnings used in deriving the requirements of the penetration analysis component.

#### 5.2.1.4.3 Covert Channel Analysis

Covert channel analysis is an assurance component that is required whenever nondiscretionary confidentiality or integrity policies are used to control information flow. It consists of (1) the identification of covert channels within a TCB, (2) the estimation of the maximum bandwidth of each channel, and (3) the testing of the covert channel handling functions.

Identifying a covert channel requires discovery of a TCB internal variable and one or more TCB interfaces that permit the alteration and viewing of variable values in violation of the information-flow policy imposed by nondiscretionary access controls. Both storage and timing channels use at least one variable for the transmission of the information being transferred between the sender and receiver. Multiple TCB interface functions may be necessary for viewing or altering a variable because after viewing or altering a variable, the sender and/or the receiver may have to set up the transmission environment for sending and/or reading the next bit. The covert channel variable may be a software, firmware, or hardware variable. In addition to TCB primitives and variables implemented by kernel and trusted processes, covert channels may use hardware-processor instructions and user-visible registers. Thus, complete covert channel analysis should take into account a product's underlying hardware architecture, not just kernels and trusted processes. Therefore, the primary goal of covert-channel identification is that of discovering all TCB variables and TCB interfaces that can be used to alter or view these variables. A secondary goal of covert channel identification is that of determining the TCB elements where time delays, noise (e.g., randomized table indices and object identifiers, spurious load), and audit code may be placed for decreasing the channel bandwidth and monitoring its use.

The term “bandwidth” is introduced to denote the rate at which information is transmitted through a channel. This use of the term bandwidth can also be related to the notion of “capacity”. The capacity of a channel is its maximum possible error-free information rate in bits per second. Thus, the primary goal of covert-channel bandwidth estimation is to determine the maximum possible error-free transmission rate, measured in bits-per-second, through a covert channel. The maximum covert channel bandwidth can be estimated using standard information theory methods. Performance measurements of the TCB are generally necessary to determine parameters required by the information theory methods.

Covert channel testing is required to demonstrate that covert channel handling functions (e.g., elimination, bandwidth limitation, audit) chosen by product designers operate correctly. Testing is also useful to confirm that the potential covert channels discovered in the product are in fact real channels. Furthermore, testing is useful when the handling functions use variable bandwidth-reduction parameters (e.g., delays) that system administrators (e.g., auditors) can set.

In contrast with maximum bandwidth estimation, which provides upper bounds for covert channels before handling functions are used, covert channel testing always requires that actual measurements be performed to determine the covert-channel bandwidths after the chosen handling functions are implemented in a product. (Of course, maximum bandwidth estimation can also be used after handling functions are implemented in a product.) Test plan documentation, including test conditions, test environment set-up, test data, expected test outcome, and actual test result documentation must be provided.

### 5.2.2 Operational Support

The operational support class of components addresses the developer's and users' responsibilities subsequent to IT product delivery. The developer's responsibilities include providing the necessary guidance to the consumer regarding the proper configuration, initialization, use, and administration of the IT product. The consumer is assumed to follow this guidance, however, fail-safe defaults may be provided to preclude consumer difficulties. An issue of concern to consumers is the identification and remediation of security flaws that may be discovered subsequent to IT product delivery. This component includes requirements for such identification and remediation.

#### 5.2.2.1 User Guidance

Requirements for user guidance help ensure that product users are able to operate the product in a secure manner (e.g., the usage constraints assumed by the protection profile must be clearly explained and illustrated). The user is defined as a person who operates the product, but has no special privileges to affect the configuration of the product. The user for most IT products is assumed to be a person with little or no computer experience, but this need not always be the case.

User's guidance is the primary means available to the developer for providing the IT product users with the necessary background and specific information on how to correctly use the product's protection functions. User guidance must do two things. First, it must explain how the protection functions of a specific product work, so that users are able to consistently and effectively protect their information. Second, it must explain the user's role in maintaining the IT product's security.

The scope of the user's guidance should be limited to documenting only the protection functions available to all users and only the responsibilities that all users have for product security. To accomplish this, the user's guidance documentation should explain what protection functions are present in the product and why, how the protection functions work, and how to use the functions properly. The material should be easy to locate in the IT product documentation and should be clear, concise, and complete.

#### 5.2.2.2 Administrative Guidance

Requirements for administrative guidance help ensure that the environmental constraints assumed by the protection profile are understood by *administrators* and *operators* of the IT product. The *administrator* is defined as a person who has the special privileges needed to affect the product configuration and set the user and product security parameters. The *operator* is defined as a person who has the special privileges needed to affect the routine operation of the product after it has been configured. The administrator has the primary responsibility for the security of the IT product. The operator is often assumed to also have some responsibility for the secure use of the IT product.

Administrative guidance is the primary means available to the developer for providing the IT product administrators with detailed, accurate information of how to: (1) configure and install an IT product, (2) operate the IT product in a secure manner, (3) make effective use of the product's privileges and protection mechanisms to control access to administrative functions and databases, and (4) avoid pitfalls and improper use of the administrative functions that would compromise the TCB and user security.

Administrator guidance should clearly illustrate necessary administrator actions (e.g., cite actual system commands and procedures). Although a high level of detail in illustrating key security concepts would benefit administrative users, the administrator guidance should not become a training manual in the areas of computer security and system administration. Administrator familiarity with the notion of IT product security should be assumed. Administrator guidance should include examples of both proper use and warnings about consequences of misuse of administrative functions, procedures, privileges, and databases. Administrator guidance should be easy to locate in the IT product documentation and should be clear, concise, and complete.

### 5.2.2.3 Flaw Remediation

Flaw remediation is an operational support assurance component for ensuring that flaws discovered by the IT product consumers will be tracked and corrected while the product is supported by the developer. While compliance with the flaw remediation requirements of a protection profile cannot be determined when a product is evaluated, it is possible to evaluate the procedures and policies that a developer has in place to track and repair flaws and distribute the repairs to affected consumers.

There are three parts to the flaw remediation process. First, the developer must be prepared to receive, validate, and track consumer reports of TCB flaws. Second, the developer must be prepared to devote resources to identifying one or more corrections to each flaw and maintaining these correction(s) with the reported flaws. Finally, the developer must have a process in place for distributing the flaw corrections to affected consumers.

### 5.2.2.4 Trusted Generation

Trusted generation is an operational support assurance component for ensuring that the copy of the IT product's TCB that is configured and activated by the consumer will exhibit the same protection properties as the master copy of the IT product's TCB that was evaluated for compliance with the protection profile. The trusted generation procedures must provide some confidence that the consumer will be aware of what product configuration parameters can affect the protection properties of the TCB. The procedures must encourage the consumer to choose parameter settings that are within the bounds assumed during the product evaluation.

### 5.2.3 Development Environment

The development environment class of components addresses the developer's engineering processes for product life cycle management, product configuration management, and trusted product distribution. These components are reviewed below.

#### 5.2.3.1 Life Cycle Definition

Life cycle definition is an assurance component for establishing that the engineering practices used by a developer to produce the IT product's TCB include the considerations and activities identified in the development process and operational support requirements of the protection profile. Consumer confidence in the correspondence between the protection profile requirements and the product's TCB is greater when security analysis and the production of evidence are done on a regular basis as an integral part of the development process and operational support activities.

The developer must explain the processes used to develop and maintain the product's TCB. The developer must also define the tools being used to analyze and implement the TCB. The higher levels of the component also require that the processes used by the developer are disciplined (i.e., consistent, measurable, and repeatable) to achieve quality products. It must be emphasized that this component imposes no constraints on the specific process chosen by the developer other than that it be sufficient to incorporate the stated requirements of the protection profile. This component simply establishes the degree of rigor required for documenting and demonstrating compliance with the developer's defined process.

#### 5.2.3.2 Configuration Management

Configuration management is an assurance component for ensuring that the IT product's TCB configuration remains consistent and complete during the product life cycle, and that changes to the TCB do not adversely affect the protection properties of the TCB. Configuration management must ensure that additions, deletions, or changes to the TCB do not compromise the correspondence between the TCB implementation and the requirements of the protection profile. This is accomplished in the configuration management component by requiring that the developer have procedures and tools that ensure that the TCB and its documentation are updated properly when the TCB changes.

Configuration management is a sound engineering practice that also provides the final element of traceability between the protection profile requirements and the product delivered to the consumer. Specifically, configuration management provides confidence that the IT product's TCB and documentation used for evaluation are the ones prepared for distribution to consumers.

The requirement of configuration management refers to four separate tasks: configuration identification, control, status accounting, and auditing. For every change that is made to the IT product, the changed version of the product, its functional requirements, and design must be identified. Control over the product configuration means that every change to the product documentation, hardware, software, or firmware is the subject of review and approval by a change-control authority. Configuration status accounting is responsible for recording and reporting on the configuration of the product throughout the change. Finally, through the process of configuration audit, the completed change can be verified to be functionally correct and consistent with the protection properties the IT product. The procedures and tools used to implement the four tasks are documented in a configuration management plan to ensure that development personnel understand their responsibilities for configuration management. Any deviation from the configuration management plan could contribute to the failure of the configuration control of an IT product and compromise the trust in the product's ability to satisfy the protection profile.

#### 5.2.3.3 Trusted Distribution

Trusted distribution is an assurance component for ensuring that the master copy of the IT product's TCB sent from the developer is the same one received by the consumer. The trusted distribution component is intended to counter the possibility that the TCB could be intentionally subverted during shipment from the development environment to the consumer.

At a minimum, the trusted distribution techniques must allow the consumer to determine if the TCB copy received has been modified during shipment. The trusted distribution techniques should also be designed to prevent any modifications from occurring during shipment.

### 5.2.4 Development Evidence

The development evidence class of components addresses requirements for the documentation of all development process, operational support, and development environment activities. The requirements for evidence are stated in four components: TCB Protection Property, Product Development, Product Analysis and Testing, and Product Support. These evidence components are elaborated below:

#### 5.2.4.1 TCB Protection Properties

The documentation of the TCB protection properties includes the definition of the functional component requirements, their modeling (if any), and their interpretation within a product's TCB.

For each functional requirement of a protection profile, a description, definition (an informal, descriptive specification), or a formal specification of the TCB components and their operation corresponding to that requirement must be provided. This correspondence must be documented to the extent necessary to establish that the functional requirements are, in fact, supported by TCB elements and interfaces. Alternate ways of presenting the evidence of this correspondence are possible. For example, the documentation may select TCB elements and interfaces, and for each individual set of selected elements and interfaces, it may identify the corresponding functional component requirement.

The correspondence between the functional component requirements and the TCB elements and interfaces can be established and documented in varying degrees of rigor. In addition to the above, the developer must document the (in)formal models of the functional component requirements, when higher levels of development assurance are desired. Providing specific models that satisfy the requirements of a profile increases the degree of rigor with which the correspondence can be established between the profile requirements and the TCB elements and interfaces. The interpretation of a model in a TCB must also be documented. However, as noted in the development assurance components, not all functional requirements must be modeled. Thus, not all aspects of this correspondence could be established at same degree of rigor. (The required modeling areas are spelled out in the assurance components.) Nevertheless, all aspects of the correspondence between the functional requirements and TCB elements and interfaces must be documented.

#### 5.2.4.2 Product Design and Implementation

The TCB design evidence includes the documentation of the (1) interface, (2) elements, (3) modular decomposition, (4) structuring support, and (5) design disciplines used. The TCB implementation evidence includes (1) the source code, and (2) the processor hardware and firmware specifications. In addition to the documentation for each stage of the development process, the design and implementation evidence should contain descriptions/definitions/specifications of the correspondences between the TCB design and the implementation.

In principle, the product design and implementation should follow a development sequence beginning with the specification of the TCB protection properties and ending with the implementation code and processor specifications. In practice, however, the different development sequences may, in fact, be executed in successive refinements, with specifications and correspondences between design and implementation being performed out of sequence. Alternative development sequences are acceptable, provided that they lead to products whose structures are accurately reflected in the design and implementation documentation.

#### 5.2.4.3 Product Testing and Analysis

The product testing and analysis evidence consists of the documentation of functional testing, penetration analysis, and covert-channel analysis.

##### 5.2.4.3.1 Functional Testing

Functional testing evidence includes test plans, test results, and test documentation. Each test plan consists of (1) the description, definition or specification of the test conditions, (2) the test data, and (3) a description of the test coverage. The test results contain the actual outcome of each test run. The test plans must be documented and, in some cases, maintained under configuration management.

##### 5.2.4.3.2 Penetration Analysis

The penetration analysis evidence includes penetration test plans and results, the documentation of the penetration testing method and tools, and when appropriate, the scenario of the discovered penetration flaws. The cause of a every discovered penetration flaw, or class of penetration flaws, must also be documented.

#### 5.2.4.3.3 Covert Channel Analysis

The covert-channel analysis evidence includes, in addition to covert-channel test plans and results, the documentation of the covert-channel identification method and tools, covert-channels found, and bandwidth estimation. All storage and timing channels found must be described in terms of the covert transmission scenarios (e.g., variables altered and viewed, source of time modulation). The cause of each covert channel, or class of covert channels, must also be documented.

### 5.2.4.4 Product Support

The product support evidence consists of the development environment and operational support documentation and tools. The development environment evidence includes the documentation of the product life-cycle process, configuration management procedures enforced, and the trusted distribution mechanisms and procedures used. This evidence also includes the identification of (1) the tools used in the product development, configuration management, and trusted distribution, and (2) the characteristics that make those tools suitable for development of protection in IT products.

The operational support evidence includes the User's Guide and the Trusted Facility Manual, the documentation describing the flaw remediation policies and procedures, and the documentation describing the trusted product generation. It also includes the description of the tools used (if any) in the product flaw remediation and trusted generation.

### 5.3 Rated Development Assurance Components

Each development assurance component addresses a unique IT development, support, or maintenance method available to an IT product developer (producer) for establishing the functional correctness of a specific product. Although these methods change over time as these disciplines mature, evolve, and new disciplines are introduced, all existing and future methods can be rated by generic characteristics. The extent to which such methods are used in a product development, maintenance, and operation can be determined by the extent to which the requirements of each method are satisfied. For this reason, the assurance components are rated according to the extent to which their requirements are satisfied. The rating of the assurance components included herein is based on the following four parameters: (1) the *scope* of the assurance method used, (2) the *precision*, or level of detail, used in, or allowed by, applying a specific method, (3) the *coverage* of the method, and (4) the *strength* of the particular method employed.

1. **Scope**. The scope of a method determines whether the method applies to all functional-component properties and to all steps of the product development, maintenance, or operation processes. For example, a specific design analysis method or a specific testing method may be applied to security-policy properties, but not to TCB protection or reference mediation properties; and a covert channel identification method and tool may apply only to the design-specification step of the development process, but not to the implementation step. Similarly, a configuration-control method may apply only to source code or to design specifications, test plans, documentation, source code, and hardware specifications; and a guidance manual (e.g., Trusted Facility Manual) referring to product operation may, or may not, include all system administration properties or requirements.

2. **Precision**. The precision in applying a method determines the level of detail at which the method is applied in product development, maintenance, or operation. For example, an analysis method may be applied to a description of a functional component, to an informal specification, or to a formal specification; it may require a formal or an informal model of the functional-component properties; it may require that formal correspondence between different levels of product design be established or only that informal correspondences be established; it may require that these correspondences show that all TCB properties are preserved by the correspondence or only that some properties are preserved. Similarly, the degree of precision in applying the method may require that the design, coding, and configuration-control methods be described or defined; that they be applied to TCB

functions, subsystems, or individual low-level modules, or only to TCB functions. Or, the degree of precision of the operational method for flaw discovery, tracking, and repair may indicate whether specific response-time deadlines are provided for flaw repair.

3. **Coverage.** The coverage of a method determines the extent to which the method is applied to a functional component, that is, whether the method is fully or only partially applied to a functional component. For example, security testing may use all test conditions required by a functional-component description or model, or only a subset of those conditions; or the test data may cover all positive and negative outcomes of a test condition or only a subset of those outcomes. Similarly, configuration management may require that all change-control conditions be applied to the configuration items or that only a subset of those conditions be applied. Or, the operational method of flaw discovery, tracking, and repair may, or may not, use all conditions of flaw discovery, tracking, and repairs, or only a subset of these conditions (e.g., use an explicit protection-problem report step, take into account the consumer protection requirements whenever protection flaws are repaired, and maintain flaw reports and corrections under configuration management).

4. **Strength.** The strength of a method used may vary according to the characteristics of the method. For example, test methods based on data flow coverage are inherently stronger than those based on boundary-value coverage (e.g., data-flow testing vs. monolithic functional testing). Covert-channel identification methods that eliminate false flows (i.e., formal flow violations) are inherently stronger than those that allow the discovery of false flows. Methods for estimating the maximum covert-channel bandwidth based on information theory are inherently stronger than those exclusively based on performance measurements. Configuration management methods and tools that automatically enforce all change-control conditions are inherently stronger than those that require operator-controlled enforcement. Compilers that enforce programming conventions and disciplines (e.g., type checking for user-defined, abstract data types) are inherently stronger than those that merely perform syntax checking.

The above parameters are chosen because, although general in nature, they facilitate the rating of the assurance components at levels of detail comparable to those of existing standards, thereby enabling potential harmonization with these standards. Other rating parameters that are equally suitable may exist. The parameters used to rate each development assurance component are summarized in Table 3.

**Table 3. Rating Summary for Development Assurance Components**

| Development Assurance Components | Scope | Precision | Coverage | Strength |
|---|---|---|---|---|
| **Development Process** | | | | |
| TCB Property Identification | | x | x | |
| TCB Design | | | | |
| TCB Element Identification | | | x | |
| TCB Interface Definition | | x | | |
| TCB Modular Decomposition | | x | x | |
| TCB Structuring Support | x | x | | |
| TCB Design Disciplines | | | x | |
| TCB Implementation | | x | x | |
| TCB Testing & Analysis | | | | |
| Functional Testing | | x | x | |
| Penetration Analysis | x | x | x | x |
| Covert Channel Analysis | x | x | x | x |
| **Operational Support** | | | | |
| User Guidance | | | | |
| Administrative Guidance | | | x | |
| Flaw Remediation | | x | x | x |
| Trusted Generation | | | x | x |
| **Development Environment** | | | | |
| Life Cycle Definition | | x | x | |
| Configuration Management | | x | x | x |
| Trusted Distribution | | | | x |
| **Development Evidence** | | | | |
| TCB Protection Properties | | x | x | |
| Product Design and Implementation | x | x | x | |
| Product Testing & Analysis | | | | |
| Functional Testing | | x | x | |
| Penetration Analysis | x | x | x | x |
| Covert Channel Analysis | x | x | x | x |
| Product Support | | x | x | x |

### 5.3.1 Development Process

#### 5.3.1.1 Rated TCB Property Identification Components

The TCB property identification components are rated based on the precision and coverage of the methods for TCB property identification. At level PD-1, the TCB properties are informally defined by interpreting the functional component requirements within the TCB. At level PD-2, precision is extended by the use of informal models of functional requirements and by requiring definitions, instead of descriptions, of the TCB element operations. At level PD-3, both the precision and coverage are extended. Precision is extended by requiring the use of formal models of functional requirements, and by the use of Descriptive Interface Specifications (DIS) for the TCB. Coverage of the interpretation method is extended by including a demonstration, by coherent arguments, that the TCB operation defined in the DIS is consistent with the appropriate formal models. At level PD-4, both precision and the coverage of the interpretation method are further extended. Precision is further extended by requiring the use of Formal Interface Specifications (FIS) for the TCB; coverage of the interpretation method is extended by including a proof that the TCB operation, as defined by the FIS, is consistent with the appropriate formal models, and by requiring that no TCB elements remain uncovered by the this interpretation.

### PD-1 Property Description

**The developer shall interpret the functional requirements of the protection profile within the product TCB. For each functional requirement, the developer shall: (1) identify the TCB elements and their TCB interfaces (if any) that implement that requirement; (2) describe the operation of these TCB elements, and (3) explain why the operation of these elements is consistent with the functional requirement.**

### PD-2 Informal Property Identification

**The developer shall provide informal models for the functional components and sub-components of the profile. At a minimum, an informal model of the access control components shall be provided. Each informal model shall include (abstract) data structures and operations defining each functional component or sub-component, and a description of the model properties. The developer shall interpret (e.g., trace) the informal models within the product TCB. For each model entity, the developer shall: (1) identify the TCB elements and their TCB interfaces (if any) that implement that entity; (2) define the operation of these TCB elements, and (3) explain why the operation of these elements is consistent with the model properties. The developer's interpretation of each informal model, which defines the TCB properties, shall identify all TCB elements that do not correspond to any model entity and shall explain why these elements do not render the TCB properties invalid.**

**For the components that are not informally modeled,** the developer shall interpret the functional requirements of the protection profile within the product TCB. For each functional requirement, the developer shall: (1) identify the TCB elements and their TCB interfaces (if any) that implement that requirement; (2) describe the operation of these TCB elements, and (3) explain why the operation of these elements is consistent with the functional requirement. **The developer's interpretation of each functional requirement, which describes the TCB properties, shall identify all TCB elements that do not correspond to any functional requirement and shall explain why these elements do not render the TCB properties invalid.**

### PD-3 Property Specification by Model Interpretation

The developer shall provide **formal** models for the functional components and sub-components of the profile. At a minimum, **a formal** model of the access control components shall be provided. **The properties of the formal models shall be clearly stated.** The developer shall **provide an interpretation of the models in the DIS of** the product's TCB. For each model entity, the developer shall: (1) identify the TCB elements and **their DIS** (if any) that implement that entity; (2) define the operation of these TCB elements, and (3) **demonstrate, by coherent arguments, that the DIS** of these elements is consistent with the model properties. The developer's interpretation of each **formal** model, which **specifies** the TCB properties, shall identify all TCB **and DIS elements (if any)** that do not correspond to any model entity and shall explain why these elements do not render the TCB properties invalid.

**An informal model of reference mediation and TCB protection shall be provided.** For the components that are **not modeled**, the developer shall interpret the functional requirements of the protection profile within the product TCB. For each functional requirement, the developer shall: (1) identify the TCB elements and their TCB interfaces (if any) that implement that requirement; (2) describe the operation of these TCB elements, and (3) explain why the operation of these elements is consistent with the functional requirement. **The developer's interpretation of each functional requirement, which describes the TCB properties, shall include all the TCB elements.**

### PD-4 Formal Specification of TCB Properties

The developer shall provide formal models for the functional components and sub-components of the profile. At a minimum, a formal model of the access control components shall be provided. The properties of the formal models shall be clearly stated. The developer shall provide a formal interpretation of the models in **the FIS** of the product's TCB. For each model entity, the developer shall: (1) identify the TCB elements and their **FIS** (if any) that implement that entity; (2) **specify** the operation of these TCB elements, and (3) **prove that the FIS** of these elements is consistent with the model properties. The developer's interpretation of each formal model, which specifies the TCB properties, shall identify all TCB and **FIS** elements (if any) that do not correspond to any model entity and shall explain why these elements do not render the TCB properties invalid.

An informal model of reference mediation and TCB protection shall be provided. For the components that are not modeled, the developer shall interpret the functional requirements of the protection profile within the product TCB. For each functional requirement, the developer shall: (1) identify the TCB elements and their TCB interfaces (if any) that implement that requirement; (2) describe the operation of these TCB elements, and (3) explain why the operation of these elements is consistent with the functional requirement. The developer's interpretation of each functional requirement, which describes the TCB properties, shall include all the TCB elements.

#### 5.3.1.2 Rated TCB Element Identification Components

The TCB element identification components are rated based on the coverage of the identification method. That is, the two levels of identification requirements are distinguished by whether the retention of protection-irrelevant elements within the TCB is justified.

**ID-1: TCB Element Identification**

**The developer shall identify the TCB elements (i.e., software, hardware/firmware code and data structures). Each element must be unambiguously identified by its name, type, release, and version number (if any).**

**ID-2: TCB Element Justification**

The developer shall identify the TCB elements (i.e., software, hardware/firmware code and data structures). Each element must be unambiguously identified by its name, type, release, and version number (if any).

**The developer shall justify the protection relevance of the identified elements (i.e., only elements that can affect the correct operation of the protection functions shall be included in the TCB). If protection-irrelevant elements are included in the TCB, the developer shall provide a rationale for such inclusion.**

### 5.3.1.3 Rated TCB Interface Definition Components

The TCB interface definition components are rated based on the precision of the interface definition method. The precision of the interface definition methods required at level IF-2 is higher than that of level IF-1, because level IF-2 requires a Descriptive Interface Specification, not just an informal interface description. Similarly the precision of the interface definition methods required at level IF-3 is higher than that of level IF-2, because level IF-3 requires a Formal Interface Specification, not just a Descriptive Interface Specification.

**IF-1: Interface Description**

**The developer shall describe all external (e.g., command, software, and I/O) administrative (i.e., privileged) and non-administrative interfaces to the TCB. The description shall include those components of the TCB that are implemented as hardware and/or firmware if their properties are visible at the TCB interface.**

**The developer shall identify all call conventions (e.g., parameter order, call sequence requirements) and exceptions signaled at the TCB interface.**

**IF-2: Interface Descriptive Specification**

The developer shall **define** all external (e.g., command, software, and I/O) administrative (i.e., privileged) and non-administrative interfaces to the TCB.

**The developer shall provide and maintain a descriptive interface specification (DIS) of the TCB that completely and accurately describes the TCB in terms of exceptions, error messages, and effects. The DIS shall identify the TCB call conventions (e.g., parameter order, call sequence requirements), and exceptions signaled. The DIS shall also include the TCB call identifier, parameter types (e.g., input, output), the effect of the call, TCB call conventions (e.g., parameter order, call sequence requirements), and exceptions handled and signaled. It shall be shown to be an accurate description of the TCB interface.**

The **DIS** shall include those components of the TCB that are implemented as hardware and/or firmware if their properties are visible at the TCB interface.

**If the TCB consists of a kernel and privileged processes, the developer shall separately identify and define the interfaces for the kernel and each privileged process.**

**Whenever covert-channel analysis, penetration analysis, and resource-constraint analysis are required, the TCB interface definition must also include all effects of a call including the direct visibility and alterability of internal TCB variables and functions.**

### IF-3: Formal Interface Specification

The developer shall define all external (e.g., command, software, and I/O) administrative (i.e., privileged) and non-administrative interfaces to the TCB.

The developer shall provide and maintain a descriptive interface specification (DIS) of the TCB that completely and accurately describes the TCB in terms of exceptions, error messages, and effects. The DIS shall identify the TCB call conventions (e.g., parameter order, call sequence requirements), and exceptions signaled. The DIS shall also include the TCB call identifier, parameter types (e.g., input, output), the effect of the call, TCB call conventions (e.g., parameter order, call sequence requirements), and exceptions handled and signaled. It shall be shown to be an accurate description of the TCB interface.

**A Formal Interface Specification (FIS) of the TCB shall be maintained that accurately describes the TCB in terms of the call identifier, parameter types (e.g., input, output), the effect of the call, TCB call conventions (e.g., parameter order, call sequence requirements), and exceptions signaled.**

The DIS **and FIS** shall include those components of the TCB that are implemented as hardware and/or firmware if their properties are visible at the TCB interface.

If the TCB consists of a kernel and privileged processes, the developer shall separately identify and define the interfaces for the kernel and each privileged process.

Whenever covert-channel analysis, penetration analysis, and resource-constraint analysis are required, the TCB interface definition must also include all effects of a call including the direct visibility and alterability of internal TCB variables and functions.

### 5.3.1.4 Rated Modular Decomposition Components

The modular decomposition components are rated based on the precision and coverage of the decomposition method. The granularity of the modular TCB decomposition at level MD-1, which delimits the precision of the decomposition method, refers to subsystem-level decomposition. The decomposition granularity is refined at level MD-2, as each subsystem is further decomposed into constituent modules. Level MD-2 also extends the coverage of the decomposition method by requiring that inter-module relationships be used in the decomposition method. Level MD-3 further extends the coverage of the decomposition method by requiring that the inter-module correctness dependencies be analyzed (see Appendix D).

### MD-1: Subsystem Decomposition

**The developer shall describe the TCB structure in terms of its design and implementation subsystems and the functional relationships between those subsystems. The developer shall identify the specific TCB protection functions (if any) associated with each subsystem and the TCB interfaces (if any) implemented by each subsystem. The developer shall describe the interfaces between the subsystems.**

**For each subsystem, the developer shall describe: the role or purpose of the subsystem, the set of related functions performed by the subsystem, and the subsystem interface (i.e., the set of invocable functions, calling conventions, parameters, global variables, and results).**

### MD-2: Module-level Decomposition

**The developer shall design the TCB as a small number (e.g., 10 to 100) of design and implementation subsystems that have well-defined functional relationships and shared-data dependencies.** The developer shall identify the specific TCB protection functions (if any) associated with each subsystem and the TCB interfaces (if any) implemented by each subsystem.

**The developer shall design each subsystem as a set of modules.** For each **module**, the developer shall describe: the role or purpose of the **module**, the set of related functions performed by the **module**, and the **module** interface (i.e., the set of invocable functions, calling conventions, parameters, global variables, and results). **The developer shall identify the protection functions of, and describe the interfaces between, these modules. The developer shall choose the modules so that the set of functions implemented by the module, the module's contribution to the TCB protection properties, and the interface(s) to the module can be described concisely (e.g., the module shall have a single purpose). The TCB structuring into modules shall be based on well-defined module relationships; for example, the contains relation (e.g., A is part of B) or the "uses" relation (e.g., A is correct only if B is correct).**

**MD-3: Module Relationship Analysis**

The developer shall design the TCB as a small number (e.g., 10 to 100) of design and implementation subsystems that have well-defined functional relationships and shared-data dependencies. The developer shall identify the specific TCB protection properties and functions associated with each subsystem and the TCB interfaces (if any) implemented by each subsystem.

The developer shall design each subsystem as a set of modules. For each module, the developer shall describe: the role or purpose of the module, the set of related functions performed by the module, and the module interface (i.e., the set of invocable functions, calling conventions, parameters, global variables, and results). The developer shall identify the protection functions of, and describe the interfaces between, these modules. The developer shall choose the modules so that the set of functions implemented by the module, the module's contribution to the TCB protection properties, and the interface(s) to the module can be described concisely (e.g., the module shall have a single purpose). The TCB structuring into modules shall be based on well-defined module relationships; for example, the contains relation (e.g., A is part of B), the "uses" relation (e.g., A is correct only if B is correct). **The developer shall analyze the correctness dependencies among these modules. This analysis may include, but is not restricted to, service and environmental dependencies.**

### 5.3.1.5 Rated TCB Structuring Support Components

The TCB structuring support components are rated based on the scope and precision of the supporting mechanisms used in TCB structuring. Ascending levels are assigned to mechanisms supporting TCB process isolation, TCB modularity, and storage objects to reflect the degrees of usefulness in TCB structuring added by these mechanisms. The precision and conceptual simplicity of these mechanisms are assigned to the highest level reflecting their importance in the rigorous analysis of TCB structuring support.

At level SP-1, the structuring of the TCB includes the minimal requirement of process isolation. Level SP-2 extends the support for TCB structuring by including the separation of protection critical elements and use of processor support for logically distinct storage objects. Level SP-3 extends the precision requirements in the definition of the protection mechanisms for TCB structuring support

**SP-1: Process Isolation**

**The TCB shall maintain process isolation.**

**SP-2: Support for Storage Objects**

The TCB shall maintain process isolation. **The TCB shall separate those elements that are protection-critical from those that are not. Features in hardware, such as segmentation, shall be used to support logically distinct storage objects with separate access-control attributes (e.g., readable, writable).**

**SP-3: Structured Protection Mechanisms**

The TCB shall maintain process isolation. The TCB shall separate those elements that are protection-critical from those that are not. Features in hardware, such as segmentation, shall be used to support logically distinct storage objects with separate access-control attributes (e.g., readable, writable). **The TCB shall employ a complete, conceptually simple, protection mechanism with precisely defined semantics. This mechanism shall play a central role in enforcing the internal structuring of the TCB and the product.**

### 5.3.1.6 Rated TCB Design Discipline Components

The TCB design discipline components are rated based on the coverage of the disciplines used for TCB structuring. The requirements range from TCB complexity minimization to the use of data hiding, layering, and high-level synchronization constructs.

At level DD-1, the design disciplines covered include that of minimizing the TCB complexity, of maximizing the use of data hiding, and of employing well-defined exception handling techniques. Level DD-2 extends this coverage by including the use of layering, high-level synchronization primitives, and multi-tasking/multi-threaded modules.

**DD-1: Specification of Disciplines Used**

**The developer shall design the product to minimize the complexity of the TCB. System engineering shall be directed towards excluding from the TCB modules that are not protection critical.**

**The TCB design shall reflect use of modern software engineering techniques, such as data hiding and abstraction (i.e., data, functional, and control abstractions) and well-defined exception-handling.**

**DD-2: Extended Disciplines for TCB Structuring**

The developer shall design the product to minimize the complexity of the TCB. System engineering shall be directed towards excluding from the TCB modules that are not protection critical.

The TCB design shall reflect use of modern software engineering techniques), such as data hiding and abstraction (i.e., data, functional, and control abstractions) and well-defined exception-handling. **The TCB design shall also include use of layering (including a rationale for each layering violation), high-level synchronization constructs, and multi-tasking/multi-threading.**

### 5.3.1.7 Rated Implementation Support Components

The implementation support components are rated according to the precision and coverage in maintaining the implementation elements of the TCB. At IM-1, the developer is only required to maintain the implementation data used to generate a physical instantiation of the TCB. IM-2 extends precision and coverage by requiring that the implementation data be organized to reflect the TCB subsystem structure and be identified as distinct configuration items. IM-3 further extends precision by requiring that the implementation data reflect the TCB module structure. Finally, IM-4 further extends the coverage of the maintenance method by requiring that the coding standards be identified and enforced, and that the implementation data modules use the same naming conventions as the design data to help establish a link between the design and the implementation.

#### IM-1: Source Data Support

**The developer shall maintain engineering diagrams and source code (as applicable) for all TCB elements.**

#### IM-2: Subsystem Correspondence Support

The developer shall maintain engineering diagrams and source code (as applicable) for all TCB elements. **The diagrams and source code for each subsystem of the TCB shall be identified and provided as configuration items.**

#### IM-3: Module Correspondence Support

The developer shall maintain engineering diagrams and source code (as applicable) for all TCB elements. The diagrams and source code for each **module** of the TCB shall be identified and provided as configuration items.

### IM-4: Naming Support For Design Correspondence

The developer shall maintain engineering diagrams and source code (as applicable) for all TCB elements. **The developer shall identify the programming languages used to develop the TCB software and reference the definitions of those languages. The developer shall identify any implementation dependent options of the programming language compiler(s) used in the TCB source code. The developer shall describe coding standards followed during the implementation of the product and shall ensure that all source code complies with these standards.** The diagrams and source code for each module of the TCB shall be identified and provided as configuration items. **The diagrams and source code shall be named using the same conventions as those used in the TCB design. The developer shall explain how the programming languages used help establish the correspondence between the TCB implementation and design.**

### 5.3.1.8 Rated Functional Testing Components

The functional testing components are rated according to the precision and coverage of the testing method. The scope of testing is constant: all functions (as represented by TCB properties) required by the protection profile must be tested. The strength of the testing method is assumed to be the same: testing is always used to show the presence of desired functionality. The precision of testing refers to the accuracy of the TCB properties and the interface definition (i.e., the interface description, DIS, or FIS) used to derive test conditions and data. The coverage of testing refers to the extent to which each function is tested (e.g., whether all or only a defined set of boundary conditions are tested).

At FT-1, the goal is to produce functional evidence that the TCB is capable of satisfying the protection profile requirements. At FT-2, the coverage of the testing is increased by requiring the tests to sample more of the range of TCB inputs. Coverage is also increased by requiring that tests for previously discovered TCB flaws be executed for all subsequent versions of the TCB (i.e., by regression testing). Precision is extended at level FT-3by requiring that interface specifications (i.e., DIS, FIS) be used to generate the test conditions and data.

### FT-1: Conformance Testing

**The developer shall test the TCB interface to show that all claimed protection functions work as stated in the TCB interface description.**

**The developer shall correct all flaws discovered by testing and shall retest the TCB until the protection functions are shown to work as claimed.**

### FT-2: TCB Interface Testing

The developer shall test the TCB interface to show that all claimed protection functions work as stated in the TCB interface description **or specification. The tests shall exercise the boundary conditions of the protection functions. The developer test procedures shall include the tests used to demonstrate the absence of all flaws discovered in previous versions of the TCB.**

The developer shall correct all flaws discovered by testing and shall retest the TCB **to show that all discovered flaws have been eliminated, no new flaws have been introduced, and the protection functions work as claimed.**

### FT-3: Specification-Driven TCB Interface Testing

The developer shall test the TCB interface to show that all claimed protection functions work as stated in the TCB interface description or specification. The tests shall exercise the boundary conditions of the protection functions. **The developer shall generate the test conditions and data from the Descriptive or Formal Interface Specification(s).** The developer test procedures shall include the tests used to demonstrate the absence of all flaws discovered in previous versions of the TCB.

The developer shall correct all flaws discovered by testing and shall retest the TCB to show that all discovered flaws have been eliminated, no new flaws have been introduced, and the protection functions work as claimed.

## 5.3.1.9 Rated Penetration Analysis Components

The penetration analysis components are rated based on the scope, precision, coverage, and strength of the analysis methods used. The scope and precision of the level PA-1 is limited to penetration testing methods referring only to unprivileged user and application programming interfaces of the TCB. The precision of penetration testing is limited to that derived from documentation of the TCB interface (e.g., system reference manuals). The coverage may be limited to the testing of known classes of penetration flaws found in other TCBs of the same, or different, types of products (e.g., generic penetration flaws).

At level PA-2, both the precision and the coverage of penetration testing are extended. The sources of design and implementation information include, in addition to system reference manuals and TCB interface description, the DIS, source code, and hardware and firmware specifications. The test conditions are systematically generated using the flaw-hypothesis method using the TCB interface specification.

Level PA-3 augments penetration testing with penetration-resistance verification methods. In particular, penetration resistance properties are defined and condition (validation) check specifications are written for each property. The DIS and source code are then verified to establish that the verification conditions are in fact implemented.

Level PA-4 represents a significant extension in the strength of the penetration analysis. That is, it requires that the penetration resistance properties of a TCB be verified formally using analysis tools. This level assumes that the design and implementation of a TCB is free of flaws that would cause penetration, and is intended to demonstrate that TCB interfaces are resistant to penetration. As such, it represents the highest level of penetration analysis assurance.

### PA-1 Basic Penetration Testing

**The developer shall define the TCB configuration, interface, and protection functions that are subject to penetration testing. For each test, the developer shall identify the goal of the test and the criteria for successful penetration. The developer shall identify all product documentation (e.g., system reference manuals) used to define penetration-test conditions, and shall document all test conditions, data (e.g., test set-up, function call parameters, and test outcomes), and coverage.**

**The penetration testing shall include, at a minimum, known classes of penetration flaws found in other TCBs (e.g., generic penetration flaws). For each uncovered flaw, the developer shall define and document scenarios of flaw exploitation, and shall identify all penetration outcomes resulting from that scenario.**

### PA-2 Flaw-Hypothesis Testing

The developer shall define the TCB configuration, interface, and protection functions that are subject to penetration testing. For each test, the developer shall identify the goal of the test and the criteria for successful penetration. **The developer shall illustrate how, in addition to system reference manuals and TCB interface description, the DIS, source code, and**

**hardware and firmware specifications are used to define penetration-test conditions. For each test, the developer shall document all test conditions, data (e.g., test set-up, function call parameters, and test outcomes), and coverage.**

**The developer shall generate the test conditions from flaw-hypotheses derived by negating assertions of TCB design capabilities and by providing counter examples that show that these assertions are false. The developer shall confirm the flaw hypotheses by checking design and implementation documentation, by defining the test data and running test programs, or by referring to known classes of penetration flaws found in other TCBs. The refutation of any hypothesis shall be documented.**

**For each uncovered flaw, the developer shall define and document scenarios of flaw exploitation and shall identify all penetration outcomes resulting from that scenario. The cause of the flaw shall be identified and documented.**

### PA-3 Penetration Analysis

The developer shall define the TCB configuration, interface, and protection functions that are subject to penetration testing **and verification**. For each test, the developer shall identify the goal of the test and the criteria for successful penetration. The developer shall illustrate how, in addition to system reference manuals and TCB interface description, the DIS, source code, and hardware and firmware specifications are used to define penetration-test conditions. For each test, the developer shall document all test conditions, data (e.g., test set-up, function call parameters, and test outcomes), and coverage.

The developer shall generate the test conditions from flaw-hypotheses derived by negating assertions of TCB design capabilities and by providing counter examples that show that these assertions are false. The developer shall confirm the flaw hypotheses by checking design and implementation documentation, by defining the test data and running test programs, or by referring to known classes of penetration flaws found in other TCBs. The refutation of each hypothesis shall be documented.

**The developer shall derive penetration-resistance properties and conditions by interpreting reference mediation and TCB protection requirements in the product's TCB. The penetration-resistance properties and conditions shall also reflect the strength of functional components (e.g., strength of the identification and authentication).**

**The developer shall verify that the penetration-resistance conditions are implemented by the TCB functions. All uncovered flaws in implementing the penetration-resistance conditions shall be documented.** For each uncovered flaw, the developer shall define and document scenarios of flaw exploitation and shall identify all penetration outcomes resulting from that scenario. The cause of the flaw shall be identified and documented.

**PA-4 Analysis of Penetration Resistance**

The developer shall define the TCB configuration, interface, and protection functions that are subject to penetration testing and verification. For each test, the developer shall identify the goal of the test and the criteria for successful penetration. The developer shall illustrate how, in addition to system reference manuals and TCB interface description, the DIS, source code, and hardware and firmware specifications are used to define penetration-test conditions. For each test, the developer shall document all test conditions, data (e.g., test set-up, function call parameters, and test outcomes), and coverage.

The developer shall generate the test conditions from flaw-hypotheses derived by negating assertions of TCB design capabilities and by providing counter examples that show that these assertions are false. The developer shall confirm the flaw hypotheses by checking design and implementation documentation, by defining the test data and running test programs, or by referring to known classes of penetration flaws found in other TCBs. The refutation of each hypothesis shall be documented.

**The developer shall use the DIS, FIS, source code, and hardware and firmware specifications to derive and specify penetration-resistance conditions, and shall document all such conditions.** The developer shall derive penetration-resistance properties and conditions by interpreting reference mediation and TCB protection requirements in the product's TCB. The penetration-resistance properties and conditions shall also reflect the strength of functional components (e.g., strength of the identification and authentication).

The developer shall verify that the penetration-resistance conditions are implemented by the TCB functions. **Tools shall be used to verify the penetration-resistance properties of the FIS and source code. The tools shall be capable of checking whether a set of penetration-resistance conditions is implemented by the FIS and/or source code of a TCB function.** All uncovered flaws in implementing the penetration-resistance conditions shall be documented. For each uncovered flaw, the developer shall define and document scenarios of flaw exploitation and shall identify all penetration outcomes resulting from that scenario. The cause of the flaw shall be identified and documented.

### 5.3.1.10 Rated Covert-Channel Analysis Components

The covert channel analysis components are rated based on the scope, precision, coverage, and strength of the analysis methods. The scope and precision of level CCA-1are limited to storage channels identified in TCB reference manuals and DIS, and the strength of maximum bandwidth estimation is limited to that provided by informal engineering measurements. The scope of identification method is increased at level CCA-2 by including both storage and timing channels and, consequently, enlarging the scope of the sources of information used (e.g., by introducing processor and hardware specifications). At level CCA-3, the precision and coverage of the covert identification are extended to include analysis of FIS and specification-to-code correspondence. Also, the strength of the maximum bandwidth estimation is increased by the requirement to use information theory methods.

**CCA-1 Analysis of Covert Storage Channels**

**1.** ***Identification:*** **The developer shall identify all sources of information used in covert-storage-channel analysis. These sources shall include TCB reference manuals and DIS. The developer shall define the identification method used. The developer shall demonstrate that the chosen identification method is sound (e.g., it leads to the discovery of all covert storage channels in the DIS or source documentation) and repeatable (i.e., independent evaluators can use the method on the same sources of covert-storage-channel information and can obtain the same results.) The developer shall define scenarios of use for each covert storage channel.**

**2.** ***Bandwidth Measurement or Engineering Estimation:*** **The developer shall define the method used for covert-storage-channel bandwidth estimation. In measuring TCB performance for covert-channel-bandwidth estimation, the developer shall satisfy the following assumptions. The maximum bandwidth estimation shall be based on the assumptions that the storage channel is noiseless, that the senders and receivers are not delayed by the presence of other processes in the product, and that the sender-receiver synchronization time is negligible. The choice of informal estimation methods shall define and justify the coding method and, therefore, the distribution of "0s" and "1s" in all transmissions.**

**The developer shall select TCB primitives to be measured for bandwidth determination from real scenarios of covert-storage-channel use. The developer shall specify TCB measurement environment for the bandwidth measurements. This specification shall include: (1) the speed of the product functions, (2) the product configuration, (3) the sizes of the memory and cache components, and (4) the product initialization. The sensitivity of the measurement results to configuration changes shall be documented. The covert-storage-channel measurements shall include the fastest TCB function calls for altering, viewing, and setting up the transmission environment; the demonstrably fastest process (context) switch time shall also be included in the bandwidth measurements. All measurements shall be repeatable.**

**3.** ***Covert Channel Testing:*** **The developer shall test all the use of all identified covert storage channels to determine whether the handling functions work as intended.**

### CCA-2 Timing Channel Analysis

1. *Identification:* The developer shall identify all sources of information used in **covert-channel** analysis. These sources shall include TCB reference manuals and DIS. **The sources of information and methods of identification shall include processor specifications whenever the identification method includes source code and hardware analysis.** The developer shall define the identification method used. The developer shall demonstrate that the chosen identification method is sound (e.g., it leads to the discovery of all **covert channels** in the DIS or source documentation) and repeatable (i.e., independent evaluators can use the method on the same sources of **covert-channel** information and can obtain the same results.) The developer shall define scenarios of use for each **covert channel**. **The developer shall also define timing channel scenarios, and shall identify all functions that provide independent sources of timing (e.g., CPUs, I/O processors).**

2. *Bandwidth Measurement or Engineering Estimation:* The developer shall define the method used for **covert-channel** bandwidth estimation. In measuring TCB performance for covert-channel-bandwidth estimation, the developer shall satisfy the following assumptions. The maximum bandwidth estimation shall be based on the assumptions that the **covert channel** is noiseless, that the senders and receivers are not delayed by the presence of other processes in the product, and that the sender-receiver synchronization time is negligible. The choice of informal estimation methods shall define and justify the coding method and, therefore, the distribution of “0s” and “1s” in all transmissions.

The developer shall select TCB primitives to be measured for bandwidth determination from real scenarios of **covert-channel** use. The developer shall specify TCB measurement environment for the bandwidth measurements. This specification shall include: (1) the speed of the product functions, (2) the product configuration, (3) the sizes of the memory and cache components, and (4) the product initialization. The sensitivity of the measurement results to configuration changes shall be documented. The **covert-channel** measurements shall include the fastest TCB function calls for altering, viewing, and setting up the transmission environment; the demonstrably fastest process (context) switch time shall also be included in the bandwidth measurements. All measurements shall be repeatable.

3. *Covert Channel Testing:* The developer shall test all the use of all identified **covert channels** to determine whether the handling functions work as intended.

### CCA-3 Formal Covert Channel Analysis

1. *Identification:* The developer shall identify all sources of information used in covert-channel analysis. These sources shall include TCB reference manuals, DIS**, and FIS**. The sources of information and methods of identification shall include processor specifications whenever the identification method includes source code and hardware analysis. The developer shall define the identification method used. The developer shall

define the identification method used. The developer shall demonstrate that the chosen identification method is sound (e.g., it leads to the discovery of all covert channels in the **FIS** or source documentation) and repeatable (i.e., independent evaluators can use the method on the same sources of covert-channel information and can obtain the same results.) **The method shall be applied on the FIS of the TCB, and shall include syntactic information-flow analysis (with or without the use of semantic analysis) or noninterference analysis. The identification of covert channels shall include specification-to-code correspondence.**

The developer shall define scenarios of use for each cover channel. The developer shall also define timing channel scenarios, and shall identify all functions that provide independent sources of timing (e.g., CPUs, I/O processors).

2. *Bandwidth Measurement or Engineering Estimation:* The developer shall define the method used for covert-channel bandwidth estimation. **The method shall be based on information theory methods.** In measuring TCB performance for covert-channel-bandwidth estimation, the developer shall satisfy the following assumptions. The maximum bandwidth estimation shall be based on the assumptions that the covert channel is noiseless, that the senders and receivers are not delayed by the presence of other processes in the product, and that the sender-receiver synchronization time is negligible.

The developer shall select TCB primitives to be measured for bandwidth determination from real scenarios of covert channel use. The developer shall specify TCB measurement environment for the bandwidth measurements. This specification shall include: (1) the speed of the product functions, (2) the product configuration, (3) the sizes of the memory and cache components, and (4) the product initialization. The sensitivity of the measurement results to configuration changes shall be documented. The covert-channel measurements shall include the fastest TCB function calls for altering, viewing, and setting up the transmission environment; the demonstrably fastest process (context) switch time shall also be included in the bandwidth measurements. All measurements shall be repeatable.

3. *Covert Channel Testing:* The developer shall test all the use of all identified covert channels to determine whether the handling functions work as intended.

### 5.3.2 Operational Support

#### 5.3.2.1 Rated User Guidance Components

The user guidance component is unrated since it contain only one level.

**UG-1: User Guide**

**The developer shall provide a User Guide which describes all protection services provided and enforced by the TCB. The User Guide shall describe the interaction between these services and provide examples of their use. The User Guide may be in the form of a summary, chapter or manual. The User Guide shall specifically describe user responsibilities. These shall encompass any user responsibilities identified in the protection profile.**

### 5.3.2.2 Rated Administrative Guidance Components

The rating of the administrative guidance components reflect, to a large degree, the rating of the security management components. At AG-1, the coverage of the Trusted Facility Manual (TFM) must include an explanation of how the TCB can be installed and used to support an organization's security policy. This explanation must include a discussion of how to set the security parameters for all TCB functions and how to use the audit trail to discover policy violations (see the administrative functions of components SM-1 and SM-2). At AG-2, TFM coverage is extended to include a discussion of how to set additional policy parameters, how to use the separate administrator and operator roles and privileges, and how to securely generate the TCB (see the administrative functions of component SM-3). Finally, at AG-3, which assumes a product with fine-grained privileges, the TFM coverage is increased to include the use of those privileges in implementing extensive administrative policies (see the administrative functions of component SM-4).

**AG-1: Basic Administrative Guidance**

**The developer shall provide a Trusted Facility Manual intended for the product administrators that describes how to use the TCB security services (e.g., Access Control, System Entry, or Audit) to enforce a system security policy. The Trusted Facility Manual shall include the procedures for securely configuring, starting, maintaining, and halting the TCB. The Trusted Facility Manual shall explain how to analyze audit data generated by the TCB to identify and document user and administrator violations of this policy. The Trusted Facility Manual shall explain the privileges and functions of administrators. The Trusted Facility Manual shall describe the administrative interaction between security services.**

**The Trusted Facility Manual shall be distinct from User Guidance, and encompass any administrative responsibilities identified in security management.**

### AG-2: Detailed Administrative Guidance

The developer shall provide a Trusted Facility Manual intended for the product administrators **and operators** that describes how to use the TCB security services (e.g., Access Control, System Entry, or Audit) to enforce a system security policy. The Trusted Facility Manual shall include the procedures for securely configuring, starting, maintaining, and halting the TCB. The Trusted Facility Manual shall explain how to analyze audit data generated by the TCB to identify and document user and administrator violations of this policy. The Trusted Facility Manual shall explain **the unique security-relevant** privileges and functions of administrators **and operators**. The Trusted Facility Manual shall describe the administrative interaction between security services.

**The Trusted Facility Manual shall identify all hardware, firmware, software, and data structures comprising the TCB. The detailed audit record structure for each type of audit event shall be described. If covert channel handling is required, the Trusted Facility Manual shall explain how to configure the product to mitigate, eliminate, or audit covert channel exploitation. The Trusted Facility Manual shall describe the cautions about and procedures for using the TCB as a base for site-specific secure applications. The Trusted Facility Manual shall describe procedures for securely regenerating the TCB after any part is changed (e.g., due to adding devices or installing flaw corrections to the TCB software).**

The Trusted Facility Manual shall be distinct from User Guidance, and encompass any administrative responsibilities identified in security management.

### AG-3: Role-Based Administrative Guidance

The developer shall provide a Trusted Facility Manual intended for the product administrators and operators that describes how to use the TCB security services (e.g., Access Control, System Entry, or Audit) to enforce a system security policy. The Trusted Facility Manual shall include the procedures for securely configuring, starting, maintaining, and halting the TCB. The Trusted Facility Manual shall explain how to analyze audit data generated by the TCB to identify and document user and administrator violations of this policy. The Trusted Facility Manual shall explain the unique security-relevant privileges and functions of administrators and operators. **The Trusted Facility Manual shall also explain the distinct security-relevant privileges and functions of the TCB and how they can be selectively granted to provide fine-grained, multi-person or multi-role system and application administration policies.** The Trusted Facility Manual shall describe the administrative interaction between security services.

The Trusted Facility Manual shall identify all hardware, firmware, software, and data structures comprising the TCB. The detailed audit record structure for each type of audit event shall be described. If covert channel handling is required, the Trusted Facility Manual shall explain how to configure the product to mitigate, eliminate, or audit covert channel exploitation. The Trusted Facility Manual shall describe the cautions about and procedures for using the TCB as a base for site-specific secure applications. The Trusted Facility Manual shall describe procedures for securely regenerating the TCB after any part is changed (e.g., due to adding devices or installing flaw corrections to the TCB software).

The Trusted Facility Manual shall be distinct from User Guidance, and encompass any administrative responsibilities identified in security management.

#### 5.3.2.3 Rated Flaw Remediation Components

The flaw remediation components are rated according to the precision, coverage and strength of the procedures used to identify and correct flaws, and disseminate corrections to affected consumers. At FR-1, the developer is responsible for establishing procedures to accept reports of flaws, find corrections to those flaws, and disseminate the flaw corrections to consumers who specifically request the corrections. At FR-2, the precision of the developer-consumer interaction is increased by requiring that the developer identify and publicize specific points of contact for product security concerns. Coverage is increased by requiring a remediation policy that distinguishes protection-relevant changes to the product from other changes. At FR-3, the coverage of both flaw repair and customer interaction procedures is increased by considering the customer's security policies and by relating each entry in the flaw tracking and repair database to the consumers who might be affected. At FR-4, precision and coverage are extended by requiring the developer to notify consumers of flaw discovery and to distribute corrections of the discovered flaws within specific time limits. Finally, at FR-5, the method is strengthened by requiring that the flaw remediation procedures be tightly coupled to the rest of the development process through the configuration management system.

**FR-1: Basic Flaw Remediation**

***Flaw Tracking Procedures*: The developer shall establish a procedure to track all reported protection flaws in each release of the product. The tracking system shall include a description of the nature and effect of each flaw and the status of finding a correction to the flaw.**

***Flaw Repair Procedures*: The developer shall establish a procedure to identify corrective actions for protection flaws.**

***Consumer Interaction Procedures*: The developer shall provide flaw information and corrections to registered consumers**.

### FR-2: Flaw Reporting Procedures

*Flaw Tracking Procedures*: The developer shall establish a procedure to track all reported protection flaws with each release of the product. The tracking system shall include a description of the nature and effect of each flaw and the status of finding a correction to the flaw.

*Flaw Repair Procedures*: The developer shall establish a procedure to identify corrective actions for protection flaws. **This procedure shall include a policy to separate protection-relevant from non-protection relevant corrections, changes, or upgrades to the product.**

*Consumer Interaction Procedures*: **The developer shall establish a procedure for accepting consumer reports of protection problems and requests for corrections to those problems. The developer shall designate one or more specific points of contact for consumer reports and inquiries about protection issues involving the product. This procedure and the designated points of contact shall be provided in the consumer documentation (e.g., the TFM or the SFUG).**

### FR-3: Systematic Flaw Remediation

*Flaw Tracking Procedures*: The developer shall establish a procedure to track all reported protection flaws with each release of the product. The tracking system shall include a description of the nature and effect of each flaw and the status of finding a correction to the flaw.

*Flaw Repair Procedures*: The developer shall establish a procedure to identify corrective actions for protection flaws. This procedure shall include a policy to separate protection-relevant from non-protection relevant corrections, changes, or upgrades to the product. **The developer shall have a policy that when a consumer's system must be used to diagnose and repair any problem, the developer personnel will abide by that consumer's system security policy.**

*Consumer Interaction Procedures*: The developer shall establish a procedure for accepting consumer reports of protection problems and requests for corrections to those problems. **This procedure shall also provide for automatic distribution of problem reports, for which corrections have been found, to registered consumers who might be affected by the problem.** The developer shall designate one or more specific points of contact for consumer reports and inquiries about protection issues involving the product. **These procedures** and the designated points of contact shall be provided in the consumer documentation (e.g., the TFM or the SFUG).

### FR-4: Timely Flaw Remediation

*Flaw Tracking Procedures*: The developer shall establish a procedure to track all reported protection flaws with each release of the product. The tracking system shall include a description of the nature and effect of each flaw and the status of finding a correction to the flaw.

*Flaw Repair Procedures:* The developer shall establish a procedure to identify corrective actions for protection flaws. This procedure shall include a policy to separate protection-relevant from non-protection relevant corrections, changes, or upgrades to the product. The developer shall have a policy that when a consumer's system must be used to diagnose and repair any problem, the developer personnel will abide by that consumer's system security policy.

*Consumer Interaction Procedures*: The developer shall establish a procedure for accepting consumer reports of protection problems and requests for corrections to those problems. This procedure shall **establish strict time intervals for automatically distributing the problem reports to registered consumers who might be affected by the problem and subsequently distributing the corrections that are found to these same consumers.** The developer shall designate one or more specific points of contact for consumer reports and inquiries about protection issues involving the product. These procedures and the designated points of contact shall be provided in the consumer documentation (e.g., the TFM or the SFUG).

## FR-5: Controlled Protection State

*Flaw Tracking Procedures:* The developer shall establish a procedure to track all reported protection flaws with each release of the product. The tracking system shall include a description of the nature and effect of each flaw and the status of finding a correction to the flaw. **The tracking system shall be incorporated into the configuration management system.**

*Flaw Repair Procedures:* The developer shall establish a procedure to identify corrective actions for protection flaws. This procedure shall include a policy to separate protection-relevant from non-protection relevant corrections, changes, or upgrades to the product. The developer shall have a policy that when a consumer's system must be used to diagnose and repair any problem, the developer personnel will abide by that consumer's system security policy.

*Consumer Interaction Procedures*: The developer shall establish a procedure for accepting consumer reports of protection problems and requests for corrections to those problems. This procedure shall establish strict time intervals for automatically distributing the problem reports to registered consumers who might be affected by the problem and subsequently distributing the corrections that are found to these same consumers. The developer shall designate one or more specific points of contact for consumer reports and inquiries about protection issues involving the product. These procedures and the designated points of contact shall be provided in the consumer documentation (e.g., the TFM or the SFUG).

#### 5.3.2.4 Rated Trusted Generation Components

The trusted generation components are rated according to the coverage and strength of the methods used to generate the baseline TCB. The goal is to produce an operational TCB that does not invalidate the protection properties established for the baseline TCB. At TG-1, the developer must provide procedures for generating an operational TCB from the delivered product. At TG-2, the coverage of the system generation method is increased by requiring the developer to have the system generation parameters default to their most restrictive settings, thereby requiring the consumer to take a positive action to reduce the protection provided by the TCB. At TG-3, the coverage and strength of the generation method are increased by requiring the developer to provide a tool that can be used after the TCB is generated to determine if the TCB parameters are within the ranges of a secure state. Finally, at TG-4, coverage and strength are further extended by requiring that the product periodically execute the parameter checking tool and alert an administrator or operator when the TCB configuration parameters are out of range.

**TG-1: Basic Trusted Generation**

> **The developer shall establish and document the procedures that a consumer must perform to generate an operational TCB from the delivered copy of the master TCB. The consumer documentation shall identify any system parameters, which are initialized or set during system generation, that affect the TCB's conformance to the protection profile and state the acceptable ranges of values for those parameters.**

**TG-2: Trusted Generation With Fail-Safe Defaults**

> The developer shall establish and document the procedures that a consumer must perform to generate an operational TCB from the delivered copy of the master TCB. The consumer documentation shall identify any system parameters, which are initialized or set during system generation, that affect the TCB's conformance to the protection profile and state the acceptable ranges of values for those parameters. **The product shall be delivered with each of these parameters set to its fail-safe defaults.**

**TG-3: Trusted Generation With Secure State Review**

> The developer shall establish and document the procedures that a consumer must perform to generate an operational TCB from the delivered copy of the master TCB. The consumer documentation shall identify any system parameters, which are initialized or set during system generation, that affect the TCB's conformance to the protection profile and state the acceptable ranges of values for those parameters. The product shall be

delivered with each of these parameters set to its fail-safe defaults. **The developer shall provide the consumer with a capability to review the product security state (e.g., by providing a program, which could be executed after generating and starting the TCB, that determines the consistency of the protection-relevant parameters).**

**TG-4: Trusted Generation With Secure State Monitoring**

The developer shall establish and document the procedures that a consumer must perform to generate an operational TCB from the delivered copy of the master TCB. The consumer documentation shall identify any system parameters, which are initialized or set during system generation, that affect the TCB's conformance to the protection profile and state the acceptable ranges of values for those parameters. The product shall be delivered with each of these parameters set to its most protective value. The developer shall provide the consumer with a capability to **monitor** the product security state (e.g., by providing a program, which is **periodically and automatically** executed after generating and starting the TCB, that determines the consistency of the protection-relevant parameters).

### 5.3.3 Development Environment

#### 5.3.3.1 Rated Life Cycle Definition Components

The life-cycle definition components are rated according to the precision and coverage of the engineering process used to develop the product. Coverage refers to the extent to which the engineering process incorporates the development and operational support requirements of a protection profile. Precision refers to the accuracy that can be brought to measuring the developer's conformance to the claimed process including the specification of the programming environment. At LC-1, the developer is required to describe the process used to develop the product, and show how all of the development and operational support requirements of the protection profile are satisfied as that process is followed. No constraints are placed on the engineering process chosen by the developer. At LC-2, the precision and coverage are extended by requiring the developer to use a well-defined process that provides for effective identification of the engineering requirements as the product is developed. Finally, at LC-3, precision and coverage are further extended by requiring a standard engineering process, which includes well-defined coding standards, whose use can be measured.

**LC-1: Developer-Defined Life Cycle Process**

**The developer shall describe the process used to develop and maintain the product. The process shall incorporate a security policy that states the technical, physical, procedural, personnel, and other measures used by the developer to protect the product and its documentation. The developer shall trace each development process and support process requirement of the protection profile to the part, or parts, of the developer's process where the requirement is satisfied. The developer shall identify the programming languages used to develop the TCB software.**

**LC-2: Standardized Life Cycle Process**

**The developer shall develop and maintain the product using a well defined, standardized engineering process.** The developer shall **explain why the process was chosen and how the developer uses it to** develop and maintain the product. The process shall incorporate a security policy that states the technical, physical, procedural, personnel, and other measures used by the developer to protect the product and its documentation. The developer shall **demonstrate that** each development process and support process requirement of the protection profile **is satisfied by some** part, or parts, of the developer's process. The developer shall identify the programming languages used to develop the TCB software **and reference the definitions of those languages. The developer shall identify any implementation dependent options of the programming language compiler(s) used to implement the TCB software.**

**LC-3: Measurable Life Cycle Process**

The developer shall develop and maintain the product using a well defined, standardized, **and measurable** engineering process. The developer shall explain why the process was chosen and how the developer uses it to develop and maintain the product. **The developer shall comply with the engineering process standard.** The process shall incorporate a security policy that states the technical, physical, procedural, personnel, and other measures used by the developer to protect the product and its documentation. The developer shall demonstrate that each development process and support process requirement of the protection profile is satisfied by some part, or parts, of the developer's process. The developer shall identify the programming languages used to develop the TCB software and reference the definitions of those languages. The developer shall identify any implementation dependent options of the programming language compiler(s) used to implement the TCB software and reference the definitions of those languages.**The developer shall describe coding standards followed during the implementation of the product and shall ensure that all source code complies with these standards.**

#### 5.3.3.2 Rated Configuration Management Components

The configuration management components are rated according to the precision, coverage, and strength of the configuration management methods. Level CM-1 includes basic configuration management methods that rely on an informal mapping between the various parts of the TCB source data, documentation, and evidence. At CM-2, the precision and strength of configuration management are increased by requiring that a rigorous mapping between configuration items be used, and that the source data configuration be controlled using automated techniques. At CM-3, coverage and strength are extended by requiring the use of a formal acceptance procedure for generating and maintaining source data. Finally, at CM-4, the strength of the overall configuration management process is enhanced by requiring that it conform to developer-defined safeguards to protect the master copy.

**CM-1: Procedural Control and Generation**

**The developer shall establish configuration control and generation procedures for developing and maintaining the TCB. The procedures shall be employed to ensure that changes to the TCB are consistent with the product's protection properties and security policy. The developer shall employ these procedures to track changes to development evidence, implementation data (e.g., source code and hardware diagrams), executable versions of the TCB, test documentation and procedures, identified flaws, and consumer documentation.**

**The configuration control procedures shall permit the regeneration of any supported version of the TCB.**

**CM-2: Automated Source Code Control**

The developer shall establish configuration control and generation procedures for developing and maintaining the TCB. The procedures shall be employed to ensure that changes to the TCB are consistent with the product's protection properties and security policy. The developer shall employ these procedures to track changes to development evidence, implementation data (e.g., source code and hardware diagrams), executable versions of the TCB, test documentation and procedures, identified flaws, and consumer documentation. **The procedures shall include automated tools to control the software source code that comprises the TCB.**

The configuration control procedures shall **assure a consistent mapping among documentation and code associated with the current version of the TCB and** permit the regeneration of any supported version of the TCB.

## CM-3: Comprehensive Automated Control

The developer shall establish configuration control and generation procedures **employing automated tools** for developing and maintaining the TCB. The procedures shall be employed to ensure that changes to the TCB are consistent with the product's protection properties and security policy. The developer shall employ these **tools** to track **and control** changes to development evidence, implementation data (e.g., source code and hardware diagrams), executable versions of the TCB, test documentation and procedures, identified flaws, and consumer documentation. **The procedures shall include a formal acceptance process for protection-relevant changes**.

The configuration control procedures shall assure a consistent mapping among documentation and code associated with the current version of the TCB and permit the regeneration of any supported version of the TCB. **The developer shall provide tools for the generation of a new version of the TCB from source code. Also, tools shall be available for comparing a newly generated version with the previous TCB version 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.**

## CM-4: Extended Configuration Management

The developer shall establish configuration control and generation procedures employing automated tools for developing and maintaining the TCB. The procedures shall be employed to ensure that **all** changes to the TCB are consistent with the product's protection properties and security policy. The developer shall employ these tools to track and control changes to development evidence, implementation data (e.g., source code and hardware diagrams), executable versions of the TCB, test documentation and procedures, identified flaws, and consumer documentation. The procedures shall include a formal acceptance process for protection-relevant changes.

The configuration control procedures shall assure a consistent mapping among documentation and code associated with the current version of the TCB and permit the regeneration of any supported version of the TCB. The developer shall provide tools for the generation of a new version of the TCB from source code. Also, tools shall be available for comparing a newly generated version with the previous TCB version 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. **The developer shall use a combination of technical, physical, and procedural safeguards to protect the master copy or copies of all material used to generate the TCB from unauthorized modification or destruction.**

#### 5.3.3.3 Rated Trusted Distribution Components

The rating of the trusted distribution components is based on the strength the trusted distribution methods; i.e., on the ability to detect or prevent modifications of the consumer's copy of the TCB from being modified while it is transferred from the development environment to the consumer's environment. At TD-1 the developer is responsible for establishing procedures and/or using technical measures that will allow a consumer to detect tampering or modification of the TCB during transfer. At TD-2, stronger methods are required to ensure that tampering with the TCB during transfer is prevented.

**TD-1: TCB Modification Detection During Distribution**

> **The developer shall establish procedures and employ appropriate technical measures to detect modifications to any TCB-related software, firmware, and hardware, including updates, that is transferred from the development environment to a consumer's site.**

**TD-2: TCB Modification Prevention During Distribution**

> The developer shall establish procedures and employ appropriate technical measures to **prevent** modifications to any TCB-related software, firmware, and hardware, including updates, that is transferred from the development environment to a consumer's site.

### 5.3.4 Development Evidence

The rating of the development evidence parallels, to a large extent, the rating of the development process, development environment, and operational support. Thus, the number of evidence levels required to reflect the process, environment and operational ratings must reflect these ratings.

The rating considerations that lead to the articulation of the development-evidence levels are similar to those used for the development process. For this reason they will not be repeated here.

### 5.3.4.1 Rated TCB Protection Property Evidence Components

**EPP-1 Evidence of TCB Correspondence to the Functional Requirements**

**The developer shall provide documentation which describes the correspondence between the functional component requirements and the TCB elements and interfaces. The TCB properties, which are defined by this correspondence, shall be explained in this documentation.**

**EPP-2 Evidence of Informal Model Interpretation in the TCB**

The developer shall provide documentation which describes the correspondence between the functional component requirements and the TCB elements and interfaces. **The developer shall also provide an informal access control model and its interpretation within the TCB.** The TCB properties, which are defined by this correspondence, shall be explained in this documentation.

**EPP-3 Evidence of Formal Model Interpretation in the DIS**

The developer shall provide documentation which describes the correspondence between the functional component requirements and the TCB elements and interfaces. **This documentation shall describe how the TCB implements the reference monitor concept.** The developer shall also provide a **formal** access-control model **and an informal reference mediation and TCB protection model**. The TCB properties, which are defined by this correspondence **and the interpretation of these models within the DIS of the TCB shall be documented by the product developer**.

**EPP-4 Evidence of Formal Model Interpretation in the FIS**

The developer shall provide documentation which describes the correspondence between the functional component requirements and the TCB elements and interfaces. This documentation shall describe how the TCB implements the reference monitor concept.The developer shall also provide a formal access-control model and an informal reference mediation and TCB protection model. The TCB properties, which are defined by this correspondence and the interpretation of these models within the DIS **and FIS** of the TCB shall be documented by the product developer.

### 5.3.4.2 Rated Product Design/Implementation Evidence Components

**EPD-1: Description Of The TCB External Interface**

**The developer shall provide an accurate description of the functions, effects, exceptions and error messages visible at the TCB interface.**

**The developer shall provide a list of the TCB elements (hardware, software, and firmware).**

**EPD-2: Specification Of The TCB External Interface**

The developer shall provide **TCB Design Specifications that include: a list of the TCB elements (hardware, software, and firmware configuration items)**; **a description of the policy allocations, functions, and interactions among the major TCB subsystems; and module level descriptions of all software and hardware in the TCB**.

**The developer shall provide a Descriptive Interface Specification (DIS) that describes the functions, effects, exceptions and error messages visible at the TCB interface. The developer shall show that the DIS is an accurate representation of the TCB's external interfaces.**

**The developer shall provide a description of the TCB's implementation and an explanation of how it corresponds to the TCB design.**

**EPD-3: Analysis Of The TCB External Interface**

The developer shall provide TCB Design Specifications that include: a list of the TCB elements (hardware, software, and firmware configuration items); **a list of protection services provided to the TCB by hardware, software, and firmware that is not part of the TCB; an explanation of the techniques and criteria used during the modular decomposition of the TCB;** a description of the policy allocations, functions, and interactions among the major TCB subsystems; and module level descriptions of all software and hardware in the TCB.

The developer shall provide a Descriptive Interface Specification (DIS) that describes the functions, effects, exceptions and error messages visible at the TCB interface. The developer shall show that the DIS is an accurate representation of the TCB's external interfaces.

**The developer shall provide TCB Implementation Data consisting of the engineering diagrams for all hardware included in the TCB and the source code used to generate the TCB software and firmware.**

**EPD-4: Policy Consistency Of The DIS**

The developer shall provide TCB Design Specifications that include: a list of the TCB elements (hardware, software, and firmware configuration items); a list of protection services provided to the TCB by hardware, software, and firmware that is not part of the TCB; an explanation of the techniques and criteria used during the modular decomposition of the TCB; a description of the policy allocations, functions, and interactions among the major TCB subsystems; and module level descriptions of all software and hardware in the TCB.

The developer shall provide a Descriptive Interface Specification (DIS) that describes the functions, effects, exceptions and error messages visible at the TCB interface **and includes a convincing argument that the DIS is consistent with the formal model of the policy**. The developer shall show that the DIS is an accurate representation of the TCB's external interfaces.

The developer shall provide TCB Implementation Data consisting of the engineering diagrams for all hardware included in the TCB and the source code used to generate the TCB software and firmware. **The developer shall show that the TCB software, firmware, and hardware implement the documented TCB design.**

**EPD-5: Policy Consistency Of The FIS**

The developer shall provide a Descriptive Interface Specification (DIS) that describes the functions, effects, exceptions and error messages visible at the TCB interface and includes a convincing argument that the DIS is consistent with the formal model of the policy. The developer shall show that the DIS is an accurate representation of the TCB's external interfaces.

**The developer shall provide a Formal Interface Specification (FIS) that rigorously defines the protection functions available at the TCB interface in terms of: the protection properties implemented by each function, the precise semantics for invoking each function, the effects of each function (i.e., returned values and effect on the TCB state), and the possible exceptions and error messages returned by each function. The FIS shall be accompanied by a convincing argument that it is consistent with the formal model of the product protection policy. This argument shall be constructed using both manual and machine-assisted specification and verification methods. Machine-assisted specification and verification methods shall be approved by the product evaluation authority.**

The developer shall provide TCB Design Specifications that include: a list of the TCB elements (hardware, software, and firmware configuration items); a list of protection services provided to the TCB by hardware, software, and firmware that is not part of the TCB; an explanation of the techniques and criteria used during the modular decomposition of the TCB; a description of the policy allocations, functions, and interactions among the major TCB subsystems; module level descriptions of all software and hardware in the TCB; **and an argument that the design implements exactly the functions specified in the FIS.**

The developer shall provide TCB Implementation Data consisting of the engineering diagrams for all hardware included in the TCB and the source code used to generate the TCB software and firmware. The developer shall show**, through either manual or machine-assisted correspondence methods,** that the TCB software, firmware, and hardware implement the documented TCB design.

### 5.3.4.3 Rated Functional Testing Evidence Components

#### EFT-1: Evidence of Conformance Testing

**The developer shall provide evidence of the functional testing that includes the test plan, the test procedures, and the results of the functional testing.**

#### EFT-2: Evidence of Test Configuration Control

The developer shall provide evidence of the functional testing that includes the test plan, the test procedures, and the results of the functional testing. **The test plans, procedures, and results shall be maintained under the same configuration control as the TCB software.**

#### EFT-3: Evidence of Specification-Driven Testing

The developer shall provide evidence of the functional testing that includes the test plan, the test procedures, and the results of the functional testing. The test, plans, procedures, and results shall be maintained under the same configuration control as the TCB software. **The test plans shall identify the TCB specification used in the derivation of the test conditions, data, and coverage analysis.**

### 5.3.4.4 Rated Penetration Analysis Evidence Components

#### EPA-1: Evidence of Penetration Testing

**The developer shall provide evidence of penetration testing. The evidence shall identify all product documentation on which the search for flaws was based. The penetration evidence shall describe the scenarios for exploiting each potential flaw in the system and the penetration test conditions, data (e.g., test set-up, function call parameters, and test outcomes), coverage, and conclusions derived from each scenario.**

**EPA-2: Evidence of Flaw-Hypothesis Generation and Testing**

The developer shall provide evidence of penetration testing. The penetration evidence shall identify all product documentation **and development evidence** on which the search for flaws was based. The penetration evidence shall describe the scenarios for exploiting each potential flaw in the system and the penetration test conditions, data (e.g., test set-up, function call parameters, and test outcomes), coverage, and conclusions derived from each scenario. **The penetration evidence shall summarize both refuted and confirmed flaws hypothesis.**

**EPA-3: Evidence of Penetration Analysis**

The developer shall provide evidence of penetration testing. The penetration evidence shall identify all product documentation and development evidence on which the search for flaws was based. The penetration evidence shall describe the scenarios for exploiting each potential flaw in the system and the penetration test conditions, data (e.g., test set-up, function call parameters, and test outcomes), coverage, and conclusions derived from each scenario. The penetration evidence shall summarize both refuted and confirmed flaws hypothesis **and identify TCB elements where the TCB implementation of the penetration-resistance conditions is flawed**.

**EPA-4: Evidence of Formal Penetration Analysis**

The developer shall provide evidence of penetration testing. The penetration evidence shall identify all product documentation and development evidence on which the search for flaws was based. The penetration evidence shall describe the scenarios for exploiting each potential flaw in the system and the penetration test conditions, data (e.g., test set-up, function call parameters, and test outcomes), coverage, and conclusions derived from each scenario. The penetration evidence shall summarize both refuted and confirmed flaws hypothesis and identify TCB elements where the TCB implementation of the penetration-resistance conditions is flawed. **The penetration evidence shall include the results of mechanically validating the implementation of the penetration resistance conditions specified for the TCB.**

### 5.3.4.5 Rated Covert Channel Analysis Evidence Components

**ECC-1: Evidence of Covert Storage Channel Analysis and Handling**

**The developer's documentation shall present the results of the covert-storage-channel analysis and the trade-offs involved in restricting these channels. All auditable events that may be used in the exploitation of known covert storage channels shall be identified. The developer shall provide the bandwidths of known covert-storage-channels whose use is not detectable by the auditing mechanism. The**

**documentation of each identified storage channel shall consist of the variable that can be viewed/altered by the channel and the TCB interface functions that can alter or view that variable. The measurements of each TCB function call used by covert-storage channels must be documented and the bandwidth computation shall be included for each channel. The measurement environment should be documented as specified. Test documentation shall include results of testing the effectiveness of the methods used to reduce covert-storage-channel bandwidths.**

#### ECC-2: Evidence of Covert Channel Analysis and Handling

The developer's documentation shall present the results of the **covert channel** analysis and the trade-offs involved in restricting these channels. All auditable events that may be used in the exploitation of known **covert channels** shall be identified. The developer shall provide the bandwidths of known **covert channels** whose use is not detectable by the auditing mechanism. The documentation of each identified **covert channel** shall consist of the **variables, timing sources, and the TCB interface functions that can be used to transmit information.** The measurements of each TCB function call used by **covert channels** must be documented and the bandwidth computation shall be included for each channel. The measurement environment should be documented as specified. Test documentation shall include results of testing the effectiveness of the methods used to reduce **covert-channel** bandwidths.

### 5.3.4.6 Rated Product Support Evidence Components

#### EPS-1: Evidence of Basic Product Support

**The developer shall provide evidence that describes the policies, procedures, and plans established by the developer to satisfy the Operational Support and Development Environment requirements of the protection profile.**

#### EPS-2: Evidence of Defined Product Support

The developer shall provide **documentation** that **defines** the policies, procedures, plans, **and tools** established by the developer to satisfy the Operational Support and Development Environment requirements of the protection profile.

**EPS-3: Evidence of Measured Product Support**

The developer shall provide documentation that defines**, explains, and justifies** the policies, procedures, plans, and tools established by the developer to satisfy the Operational Support and Development Environment requirements of the protection profile. **The documentation shall also explain how the developer periodically evaluates compliance with the established procedures, policies, and plans.**

## 5.4 Bibliographic Notes

TBD.

# Chapter 6.

# EVALUATION ASSURANCE REQUIREMENTS

*Editor's Note: This chapter represents an initial attempt to consolidate many different ideas regarding evaluations and articulate a simple structure for levying requirements on the evaluation process. The material is presented to stimulate the debate and analysis regarding what should be required of product evaluations.*

## 6.1 Overview

Product evaluation is the process of validating that an IT product, and the context in which it is developed and supported, conforms to the requirements of a protection profile. Since only the protection functions and quality of the product mitigate against risk, the consumer's understanding of residual risk in any system employing the product is largely dependent upon a producer's claims and/or upon product evaluation information. Quality, in this context, is focused on appropriateness, correctness, and simplicity of design with respect to functional requirements, and correctness, effectiveness, and efficiency of implementation with respect to design. When this information is provided by a source independent of the product's producer, the consumer generally has a greater degree of confidence regarding the degree of conformance claimed by the producer.

This chapter addresses the protection profile section for evaluation assurance which contains requirements derived from the generic components presented later in this chapter. These generic requirements may be tailored with respect to the profile requirements for protection functions and development assurance. Each protection profile can be separately tailored for evaluation. Thus, all IT products produced to conform to a particular protection profile will be commonly evaluated at a level of assurance commensurate with the profile's requirements for protection functions and development assurance. This evaluation assurance level is agreed upon during profile registration by the participants to the registration process (e.g., producers, profile developers, evaluation authorities).

The evaluation assurance requirements contained in a protection profile specify the *minimum* requirements that must be satisfied during an evaluation process. This document adopts the philosophy that if a protection function or development assurance requirement is placed on a producer, then the satisfaction of such a requirement must be evaluated. Incremental evaluation assurance is accomplished by changing the scope and intensity of examination to make the evaluated aspects of the product's TCB, its internals, its interfaces, and its production processes increasingly visible.

Evaluation assurance requirements do not by themselves define a particular approach to product evaluation. There are conceivably many different approaches to product evaluation to provide varying levels of assurance. Any approach is defined by both evaluation methods and the business process that incorporates those methods. Since the business process is one that should remain flexible, the requirements specified in this document are not intended to completely define a specific process. Rather, they articulate requirements on methods that can be used with a variety of business processes.The specific process is largely the result of business decisions made by an evaluation authority, often in conjunction with the producer and/or consumer, regarding the most appropriate and cost-effective manner to accomplish the evaluation assurance goals within the available resources.

This chapter is divided into four sections. The remainder of this section groups the evaluation assurance components of a TCB into three classes and describes the types of components in each class. The second section presents a description of each type of evaluation assurance component in terms of the functional and development assurance requirements these components are intended to verify. The third section presents the rated evaluation assurance components. The last section includes a bibliography of useful literature references.

**Classes of Evaluation Assurance:** The product evaluation components address three classes of evaluation methods (i.e., *testing, review,* and *analysis*) and establish generic evaluation requirements based on those methods. Test analysis and independent testing were grouped due to the similarity of their requirements. The product evaluation components are depicted in Figure 6.

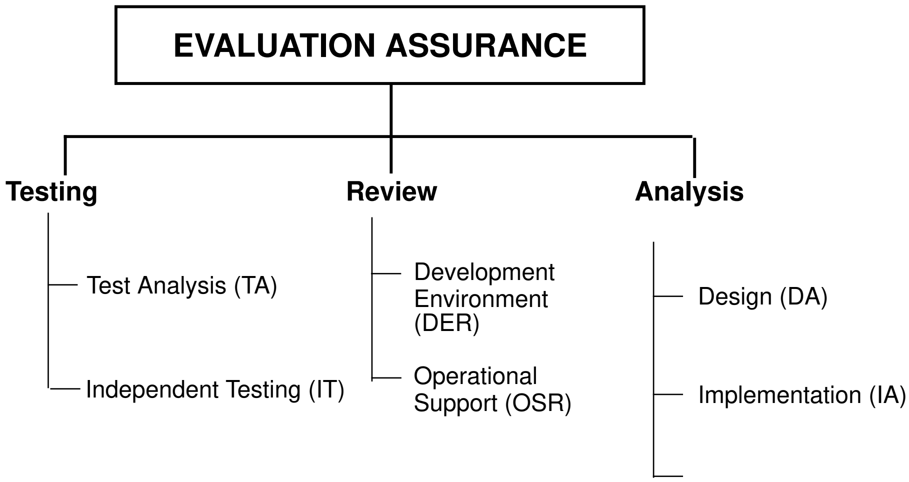



**Figure 6. Taxonomy of Evaluation Assurance Components.**

**Testing.** This class of components defines two evaluation assurance components: (1) test analysis components, and (2) independent testing components. These two components determine whether the product's TCB meets the functional protection requirements as defined in the functional requirements section of the protection profile. These components also assess whether activities required for TCB property definition and TCB testing & analysis (both found in the development process section of development assurance component section of the protection profile) verification have been accomplished. These components further assess whether these activities have been documented in accordance with the development evidence requirements of the development assurance section of the profile.

**Review.** This class of components defines two evaluation assurance components: (1) development environment components, and (2) operational support components. These two components validate compliance with the operational support and development support aspects of the development assurance requirements section of the protection profile.

**Analysis.** This class of components defines two evaluation assurance components: (1) design analysis components and (2) implementation analysis components. These two components validate compliance with the TCB design and TCB implementation support aspects of the development assurance requirements section of the protection profile.

## 6.2 Evaluation Assurance Components

*Editor's Note: The components included in this section are provided to serve as examples and are, in some cases, incomplete. Comments regarding their structure and content are desired from all reviewers. An effort was made to concentrate only on evaluation requirements and to exclude any process-oriented areas (though some process-oriented implications may remain). The requirement for specifying these components is to make them generic (i.e., suitable for a variety of evaluation processes) and to be able to place them in context with the other profile requirements (i.e., evaluation requirements should be commensurate with the functional requirements and development assurance requirements).*

### 6.2.1 Testing

Evaluation testing requirements will apply in all protection profiles. Testing of the product and its protection functions is a responsibility of the producer. The producer may also have the product beta-tested by independent sources. Evaluation testing includes (1) the analysis of the appropriateness, coverage, consistency and completeness of the beta-test site's test suite and/or the producer's test suite, the data resulting from conducting these tests, and (2) the independent application and analysis of testing by the evaluation team. The evaluation process will be required to assess the producer's testing results and may be required to independently perform some level of testing of the product. An example of such an evaluation testing requirement would be where the evaluation team must execute the producer's functional tests and then re-execute them after any discovered errors (either with the tests or the product) have been corrected.

Evaluation testing may be as simple as a pass or fail conformance test suite against which the products must be tested. For more comprehensive functional testing, the evaluators may be required to functionally test aspects of the product not covered by the producer's testing. The product's TCB, with respect to its ability to resist penetration, will also require a range of penetration analysis and testing. Such testing begins with known generic flaws and proceeds to hypotheses that are refuted or confirmed. Again, the evaluators may analyze the product's tests and test results, rerun all or a selected set of such tests, or develop additional tests not covered by other testing. If covert channel handling methods are incorporated into a product to limit bandwidth, the effectiveness of those methods in reducing channel bandwidths must also be tested. In general, the more robust and/or resilient the product's protection is expected to be, the more significant the level of testing that should be performed.

#### 6.2.1.1 Test Analysis Components

Test analysis establishes the testing analysis requirements needed to determine whether the product meets the functional protection requirements as defined in the protection profile. The producer will always perform this functional testing. Functional testing is based on the operational product, the TCB's functional properties, the product's operational support guidance, and other producer's documentation as defined by the development evidence requirements. Functional test analysis is based on the achieved test results as compared to the expected results derived from the development evidence. Penetration test analysis is based on known generic penetration flaws and a set of flaw hypotheses established for the specific product implementation. Covert channel bandwidth testing is based on the bandwidth prior to the application of covert channel handling method and the bandwidth that results after such application.

#### 6.2.1.2 Independent Testing Components

Independent testing establishes the testing requirements performed by a testing agent not associated with the producer. These requirements determine whether the product's TCB meets the functional protection requirements as defined in the protection profile. Testing is based on the operational product, the TCB's functional properties, the product's operational support guidance, and other producer's documentation as defined by the development evidence requirements.

### 6.2.2 Evaluation Review Requirements

This aspect of evaluation assurance addresses validating compliance with the development assurance requirements. Evaluation reviews simply check for a process, discipline, or form of documentation by examining evidence that validates presence or absence. Two aspects of compliance are reviewed; (1) compliance with development environment requirements and (2) compliance with operational support requirements.

#### 6.2.2.1 Development Environment Review

The development environment review establishes the level of review required to determine whether the product meets the requirements as defined in the protection profile's development assurance subsections for development environment. This includes the components life-cycle definition, configuration management, and trusted distribution. An example of such review would be configuration management audits performed by the evaluation team to ensure that a configuration management plan is being properly applied. At a certain level, the evaluation team must conduct a configuration audit of all the software, firmware, and hardware required to be kept under configuration control according to the (approved) configuration management plan. Similar requirements would apply to trusted distribution and life cycle management.

#### 6.2.2.2 Operational Support Review

The operational support review establishes the level of review required to determine whether the product meets the requirements as defined in the protection profile's development assurance subsections for operational support. This includes the components for user and administrative guidance, flaw discovery, tracking, and repair procedures, and trusted generation.

### 6.2.3 Evaluation Analysis Requirements

This aspect of evaluation assurance addresses validating compliance with two aspects of the development assurance requirements. Analysis requirements are established to determine whether the product meets the development assurance requirements. The analysis is based on the producer's documentation, as defined by the development evidence requirements. The two aspects analyzed are: (1) compliance with TCB design requirements and (2) compliance with TCB implementation support requirements.

#### 6.2.3.1 Design Analysis

Design analysis requirements specify the objectives for evaluating a product from a design perspective (i.e., without examination of the product implementation). These requirements also address the adequacy of required design documentation. A design analysis may range from a relatively simple functional overview (e.g., a black-box perspective of the TCB) to a detailed analysis of internal design details, modularity, layering, etc. The level of evaluation analysis required for producer-supplied documentation will be commensurate with the product's design requirements as set forth in the protection profile's development assurance section.

#### 6.2.3.2 Implementation Analysis

Implementation analysis requirements address areas such as code analysis. An example of such analysis is a requirement wherein the evaluation team must examine at least 50% of the TCB's code to ascertain whether the TCB meets the modularity requirements. The selected code must be a representative set of the TCB and (as appropriate) include samples of code from several different programmers.

## 6.3 Rated Evaluation Assurance Components

### 6.3.1 Rated Test Analysis Components

This component establishes the testing analysis requirements to determine whether the product meets the functional protection requirements as defined in the protection profile. This component is required for all evaluations as it assumes that the producer will always perform functional testing.

**TA-1: Elementary Test Analysis**

> **The evaluator shall assess whether the producer has performed the activities defined in the development assurance requirements of the protection profile for functional testing and whether the producer has documented these activities as defined in the development evidence requirements of the protection profile. The evaluator shall analyze the results of the producer's testing activities for completeness of coverage and consistency of results. The evaluator shall determine whether the product's protection properties, as described in the product documentation have been tested. The evaluator shall assess testing results to determine whether the product's TCB works as claimed.**

**TA-2: Enhanced Test Analysis**

The evaluator shall assess whether the producer has performed the activities defined in the development assurance requirements of the protection profile for functional testing **and penetration analysis,** and whether the producer has documented these activities as defined in the development evidence requirements of the protection profile. The evaluator shall analyze the results of the producer's testing activities for completeness of coverage and consistency of results, **and general correctness (e.g., defect trend from regression testing). This analysis shall examine the testability of requirements, the adequacy of the tests to measure the required properties, the deviation of the actual results obtained from the expected results, and a general interpretation of what the testing results mean.** The evaluator shall determine whether the product's protection properties, as described in the product documentation, **and all relevant known penetration flaws** have been tested. The evaluator shall assess testing results to determine whether the product's TCB works as claimed**, and whether there are any remaining obvious ways (i.e., ways that are known, or that are readily apparent or easily discovered in product documentation) for an unauthorized user to bypass the policy implemented by the TCB or otherwise defeat the product's TCB.**

**TA-3: Extended Test Analysis**

The evaluator shall assess whether the producer has performed the activities defined in the development assurance requirements of the protection profile for functional testing and penetration analysis, and whether the producer has documented these activities as defined in the development evidence requirements of the protection profile. The evaluator shall analyze the results of the producer's testing activities for completeness of coverage and consistency of results, and general correctness (e.g., defect trend from regression testing). This analysis shall examine the testability of requirements, the adequacy of the tests to measure the required properties, the deviation of the actual results obtained from the expected results, and a general interpretation of what the testing results mean. The evaluator shall determine whether the product's protection properties**, as defined at the TCB interface (i.e., by the DIS),** and all relevant known penetration flaws have been tested. **The evaluator shall independently develop, test, and document additional flaw hypotheses.** The evaluator shall assess testing results to determine whether the product's TCB works as claimed, **that the TCB's implementation is consistent with the DIS,** and whether there are any obvious ways (i.e., ways that are known, or that are readily apparent or easily discovered in product documentation) for an unauthorized user to bypass the policy implemented by theTCB or otherwise defeat the product's TCB, **and whether all discovered TCB flaws have been corrected and no new TCB flaws introduced. The evaluator shall determine whether the product is relatively resistant to penetrations.**

### TA-4: Comprehensive Test Analysis

The evaluator shall assess whether the producer has performed the activities defined in the development assurance requirements of the protection profile for functional testing and penetration analysis, and whether the producer has documented these activities as defined in the development evidence requirements of the protection profile. The evaluator shall analyze the results of the producer's testing activities for completeness of coverage and consistency of results, and general correctness (e.g., defect trend from regression testing). This analysis shall examine the testability of requirements, the adequacy of the tests to measure the required properties, the deviation of the actual results obtained from the expected results. **The analysis shall extend to trace all defects identified, corrected, and retested. The analysis shall include an assessment of test coverage and completeness, and defect frequency. The results of testing shall be interpreted in terms that express product performance and protection adequacy.** The evaluator shall determine whether the product's protection properties, **as defined for all protection-relevant modules of the TCB,** and all relevant known penetration flaws have been tested. The evaluator shall independently develop, test, and document additional flaw hypotheses. The evaluator shall assess testing results to determine whether the product's TCB works as claimed, that the TCB's implementation is consistent with the DIS, and whether there are any obvious ways (i.e., ways that are known, or that are readily apparent or easily discovered in product documentation) for an unauthorized user to bypass the policy implemented by theTCB or otherwise defeat the product's TCB, and whether all discovered TCB flaws have been corrected and no new TCB flaws introduced. **No design flaws and no more than a few correctable implementation flaws may be found during testing and there shall be reasonable confidence that few remain. If covert channel handling methods have been implemented, the testing results shall show that the methods used to reduce covert channel bandwidths have been effective for all evaluated configurations.** The evaluator shall determine whether the product is relatively resistant to penetrations.

### TA-5: Formal Test Analysis

The evaluator shall assess whether the producer has performed the activities defined in the development assurance requirements of the protection profile for functional testing and penetration analysis, and whether the producer has documented these activities as defined in the development evidence requirements of the protection profile. The evaluator shall analyze the results of the producer's testing activities for completeness of coverage and consistency of results, and general correctness (e.g., defect trend from regression testing). This analysis shall examine the testability of requirements, **use of the FIS for test derivation,** the adequacy of the tests to measure the required properties, the deviation of the actual results obtained from the expected results. The analysis shall extend to trace all defects identified, corrected, and retested. The analysis shall include an assessment of test coverage and completeness, and defect frequency. The results of testing shall be interpreted in terms that express product performance and protection adequacy. The evaluator shall determine whether the product's protection properties**, as defined for the entire TCB,** and all relevant known penetration flaws have been tested. The evaluator shall independently develop, test, and document additional flaw hypotheses. The evaluator shall assess testing results to determine whether the product's TCB works as claimed**, that the TCB's implementation is consistent with the FIS**, and whether there are any obvious ways (i.e., ways that are known, or that are readily apparent or easily discovered in product documentation) for an unauthorized user to bypass the policy implemented by theTCB or otherwise defeat the product's TCB, and whether all discovered TCB flaws have been corrected and no new TCB flaws introduced. No design flaws and no more than a few correctable implementation flaws may be found during testing and there shall be reasonable confidence that few remain. If covert channel handling methods have been implemented, the testing results shall show that the methods used to reduce covert channel bandwidths have been effective for all evaluated configurations. The evaluator shall determine whether the product is **completely resistant** to penetrations.

### 6.3.2 Rated Independent Testing Components

This component establishes the independent testing requirements to determine whether the product's TCB meets the functional protection requirements as defined in the protection profile.

#### IT-1: Elementary Independent Testing

**A tester, independent of the producer or evaluator, shall perform functional and elementary penetration testing. This testing shall be based on the product's user and administrative documentation, and on relevant known penetration flaws. Satisfactory completion consists of demonstrating that all user-visible security enforcing functions and security-relevant functions work as described in the product's user and administrative documentation and that no discrepancies exist between the documentation and the product. Test results of the producer shall be confirmed by the results of independent testing. The evaluator may selectively reconfirm any test result.**

**If the independent testing is performed at beta-test sites, the producer shall supply the beta-test plan and the test results. The evaluator shall review the scope and depth of beta testing with respect to the required protection functionality, and shall verify independence of both the test sites and the producer's and beta-test user's test results. The evaluator shall confirm that the test environment of the beta-test site(s) adequately represents the environment specified in the protection profile.**

### IT-2: Enhanced Independent Testing

**The evaluator shall independently perform functional and elementary penetration testing to confirm test results.** This testing **may be selective and** shall be based **on (1) the results of other independent and/or producer testing, (2) the TCB's DIS, (3) other product design and implementation documentation, (4)** the product's user and administrative documentation, and (5) relevant known penetration flaws. **Satisfactory completion consists of demonstrating that all TCB functions work as described in the product's relevant documentation, that test results are consistent,** and that no discrepancies exist between the documentation and the product.

If the independent testing is performed at beta-test sites, the producer shall supply the beta-test plan and the test results. The evaluator shall review the scope and depth of beta testing with respect to the required protection functionality, and shall verify independence of both the test sites and the producer's and beta-test user's test results. The evaluator shall also confirm that the test environment of the beta-test site(s) adequately represents the environment specified in the protection profile.

### IT-3: Comprehensive Independent Testing.

The evaluator shall independently perform functional and elementary penetration testing to confirm test results. This testing may be selective and shall be based on (1) the results of other independent and/or producer testing, (2) the TCB's DIS, (3) other product design and implementation documentation, (4) the product's user and administrative documentation, (5) relevant known penetration flaws**, and (6) evaluator-developed TCB penetration flaw hypotheses and corresponding tests that attempt to exploit the hypothesized flaws.** Satisfactory completion consists of demonstrating that all TCB functions work as described in the product's relevant documentation, that test results are consistent, and that no discrepancies exist between the documentation and the product. **Satisfactory penetration test completion shall be determined by the subjective judgement (which may be supported algorithmically) of the evaluator. Test duration agreements may further constrain this judgement. Categorization of an actual penetration flaw shall be based on the reproducibility of that flaw. Flaws that are discovered, but are not reproducible shall remain categorized as potential penetration flaws. All actual penetration flaws must be corrected and retested.**

**The evaluator shall provide a penetration test plan document that describes the additional evaluator-developed flaw hypotheses and associated tests. The evaluator shall execute these tests and shall report any discovered flaws to the producer as part of the testing results. At the conclusion of penetration testing, the evaluator shall provide copies of this penetration test plan and its test results to the producer. The producer shall ensure that this test plan and its test results are incorporated into the rest of the product's testing documentation and that such documentation is available for further analysis throughout the life of the product.**

**If the product has incorporated covert channel handling, the evaluator shall test for covert channel bandwidth reductions to determine the effectiveness of handling method(s) in reducing the bandwidths of identified covert channels for all evaluated configurations.**

If the independent testing is performed at beta-test sites, the producer shall supply the beta-test plan and the test results. The evaluator shall review the scope and depth of beta testing with respect to the required protection functionality, and shall verify independence of both the test sites and the producer's and beta-test user's test results. The evaluator shall also confirm that the test environment of the beta-test site(s) adequately represents the environment specified in the protection profile.

## IT-4: Formal Independent Testing.

The evaluator shall independently perform functional and elementary penetration testing to confirm test results. This testing shall be based on (1) the results of producer or other independent testing, (2) **the TCB's FIS,** (3) the product's design and implementation documentation, (4) the product's user and administrative documentation, (5) relevant known penetration flaws, and (6) evaluator-developed TCB penetration flaw hypotheses and corresponding tests that attempt to exploit the hypothesized flaws. Satisfactory completion consists of demonstrating that all TCB functions work as described in the product's **relevant** documentation, **that the TCB functions are consistent with the FIS,** that test results are consistent, and that no discrepancies exist between the documentation and the product. Satisfactory penetration test completion shall be determined by the subjective judgement of the evaluator (which may be supported algorithmically). Test duration agreements may further constrain this judgement. Categorization of an actual penetration flaw shall be based on the reproducibility of that flaw. Flaws that are discovered, but are not reproducible shall remain categorized as potential penetration flaws. All actual penetration flaws must be corrected and retested.

The evaluator shall provide a penetration test plan document that describes the additional evaluator-developed flaw hypotheses and associated tests. The evaluator shall execute these tests and shall report any discovered flaws to the producer as part of the testing results. At the conclusion of penetration testing, the evaluator shall provide copies of this penetration test plan and its test results to the producer. The producer shall ensure that this test plan and its test results are incorporated into the rest of the product's testing documentation and that such documentation is available for further analysis throughout the life of the product.

If the product has incorporated covert channel handling, the evaluator shall test for covert channel bandwidth reductions to determine the effectiveness of handling method(s) in reducing the bandwidths of identified covert channels.

If the independent testing is performed at beta-test sites, the producer shall supply the beta-test plan and the test results. The evaluator shall review the scope and depth of beta testing with respect to the required protection functionality, and shall verify independence of both the test sites and the producer's and beta-test user's test results. The evaluator shall also confirm that the test environment of the beta-test site(s) adequately represents the environment specified in the protection profile.

### 6.3.3 Rated Development Environment Review Components

This component establishes the level of review required to determine whether the product meets the requirements as defined in the protection profile’s development assurance subsections for development environment including life-cycle definition and configuration management, and trusted distribution.

**DER-1: Elementary Development Environment Review**

**The evaluator shall review the producer's development and maintenance process description documentation to determine the degree of discipline enforced upon and within the process, and to determine the protection characteristics associated with the product's development and maintenance. The results of this review shall establish, for the evaluator, the producer’s development environment, its policies, and the degree of enforcement maintained during development execution.**

**DER-2: Enhanced Development Environment Review**

The evaluator shall review the producer's development and maintenance process description documentation **and shall conduct a random audit of the producer's processes using the evidence generated by each process** to determine the degree of discipline enforced upon and within the process, and to determine the protection characteristics associated with the product's development and maintenance. The results of this review shall establish, for the evaluator, the producer’s development environment, its policies, and the degree of enforcement maintained during development execution. **The results of this review shall also confirm the producer’s general conformance with relevant development environment requirements.**

**DER-3: Comprehensive Development Environment Review**

The evaluator shall review the producer's development and maintenance process description documentation and shall conduct a **complete** audit of the producer's processes using the evidence generated by each process to determine the degree of discipline enforced upon and within the process, and to determine the protection characteristics associated with the product's development and maintenance. The results of this review shall establish, for the evaluator, the producer's development environment, its policies, and the degree of enforcement maintained during development execution. The review shall also confirm the producer's **complete conformance with all relevant development environment requirements.**

### 6.3.4 Rated Operational Support Review Components

This component establishes the level of review required to determine whether the product meets the requirements as defined in the protection profile's development assurance subsections for operational support including user and administrative guidance, flaw discovery, tracking, and repair procedures, and trusted generation.

#### OSR-1 Elementary Operational Support Review

**The evaluator shall review all documentation focused on the activities of product use (e.g., Users Manuals) and product administration including installation, operation, maintenance, and trusted recovery (e.g., Trusted Facility Management Manuals). This review shall assess the clarity of presentation, difficulty in locating topics of interest, ease of understanding, and completeness of coverage. The need for separate manuals dedicated to protection-relevant aspects of the product should be assessed for effectiveness.**

This component should also address flaw remediation and trusted generation. [[TBD.]]

#### OSR-2 Enhanced Operational Support Review

The evaluator shall review all documentation focused on the activities of product use (e.g., Users Manuals) and product administration including installation, operation, maintenance, and trusted recovery (e.g., Trusted Facility Management Manuals). This review shall assess the clarity of presentation, difficulty in locating topics of interest, ease of understanding, and completeness of coverage. The need for separate manuals dedicated to protection-relevant aspects of the product should be assessed for effectiveness. **The evaluator shall randomly select a sample of the documented protection-relevant features and procedures and execute them to determine if their descriptions are accurate and correct.**

This component should also address flaw remediation and trusted generation. [[TBD.]]

**OSR-3 Comprehensive Operational Support Review**

The evaluator shall review all documentation focused on the activities of product use (e.g., Users Manuals) and product administration including installation, operation, maintenance, and trusted recovery (e.g., Trusted Facility Management manuals. This review shall assess the clarity of presentation, difficulty in locating topics of interest, ease of understanding, and completeness of coverage. The need for separate manuals dedicated to protection-relevant aspects of the product should be assessed for effectiveness. The evaluator shall **execute all documented protection-relevant features and procedures to determine if their descriptions are accurate and correct.**

This component should also address flaw remediation and trusted generation. [[To be written.]]

### 6.3.5 Rated Design Analysis Components

This component establishes the analysis requirements to determine whether the product meets the design requirements as defined in the development process assurance section of the protection profile, including the TCB property definition and TCB design requirements. The analysis is based on the producer's documentation, as defined by the development evidence requirements.

**DA-1: Elementary Design Analysis**

**The evaluator shall determine whether the producer has performed the activities defined in the development process assurance requirements of the protection profile for TCB property definition and TCB design. The evaluator shall determine whether the producer has documented these activities as defined in the development evidence requirements of the protection profile. The evaluator shall analyze the results of the producer's activities for completeness and consistency of design with respect to requirements.**

**DA-2: Enhanced Design Analysis**

The evaluator shall determine whether the producer has performed the activities defined in the development process assurance requirements of the protection profile for TCB property definition and TCB design. The evaluator shall determine whether the producer has documented these activities as defined in the development evidence requirements of the protection profile. The evaluator shall analyze the results of the producer's activities for completeness, consistency, **and correctness** of design with respect to requirements.

**DA-3: Comprehensive Design Analysis**

The evaluator shall determine whether the producer has performed the activities defined in the development process assurance requirements of the protection profile for TCB property definition and TCB design. The evaluator shall determine whether the producer has documented these activities as defined in the development evidence requirements of the protection profile. The evaluator shall analyze, **with the help of formal methods and appropriate automated tools,** the results of the producer's activities for completeness, consistency, and correctness of design with respect to requirements **(e.g., validating the formal verification of the design).**

### 6.3.6 Rated Implementation Analysis Components

This component establishes the implementation analysis required to determine whether the product meets the requirements as defined in the TCB implementation requirements in a protection profile’s development assurance section. The analysis is based on the implemented code and on the producer's documentation, as defined by the development evidence requirements.

**CI-1: Elementary Implementation Analysis**

**The evaluator shall conduct a code inspection on a small sample of randomly selected product code. The assessment shall focus on clarity of the coding style, adherence to coding standards, coding documentation, and on possible software defects that may be present with respect to the product’s informal design. The inspection shall be performed to obtain only a sample of possible software defects, not to capture all such possible defects. The evaluator shall report all discovered defects to the producer; the assessment shall report the number of defects found per line of code inspected from the random sample size. Use of producer-provided code inspection results can supplement this sample inspection. All trapdoors built into the product for maintenance purposes shall be identified by the producer and shown to be protected by the product.**

**CI-2: Enhanced Implementation Analysis**

The evaluator shall conduct a code inspection on a **moderate** sample of randomly selected product code. The assessment shall focus on clarity of the coding style, adherence to coding standards, coding documentation, and on possible software defects that may be present with respect to the product’s informal design **and model**. The inspection shall be performed to obtain only a sample of possible software defects, not to capture all such possible defects. The evaluator shall report all discovered defects to the producer; the assessment shall report the number of defects found per line of code inspected from the random sample size. Use of producer-provided code inspection results can supplement this sample inspection. All trapdoors built into the product for maintenance purposes shall be identified by the producer and shown to be protected by the product.

### CI-3: Comprehensive Implementation Analysis

The evaluator shall conduct an inspection on a moderate sample of randomly selected product code. The assessment shall focus on the clarity of the coding style, adherence to coding standards, coding documentation, and on possible software defects that may be present with respect to the product's **formal** design and model. The inspection shall be performed to obtain only a sample of possible software defects, not to capture all such possible defects. The evaluator shall report all discovered defects to the producer; the assessment shall report the number of defects found per line of code inspected from the random sample size. Use of producer-provided code inspection results can supplement this inspection. All trapdoors built into the product for maintenance purposes shall be identified by the producer and shown to be protected by the product. **The producer shall correct all discovered defects and the corrected software reinspected. A rigorous analysis of the implementation's correspondence to the verified design shall be performed by the evaluator to validate correctness. Such analysis may be supported by appropriate automated tools.**

## 6.4 Bibliographic Notes

TBD.

# Chapter 7.

# CONSTRUCTION OF PROTECTION PROFILES

## 7.1 Overview

The functional and assurance components and their ratings defined in previous chapters provide the basic building blocks for the definition of protection profiles. The definition of a protection profile consists of assembling different functional and assurance components into a consistent and coherent set that satisfies specific security goals of the anticipated environments of product use. The assembled components and their requirements are generally intended to counter threats, eliminate vulnerabilities, support security standards, and satisfy regulatory requirements defined in the anticipated environments of use.

During profile construction, environment-specific requirements are used to select and synthesize the functional and the assurance components for IT product development (see Chapter 3). It should be noted that not all environment-specific requirements are relevant to the selection of the functional and assurance components. For example, some environment-specific requirements may address only problems of organization management and IT product use that have no direct impact on IT product requirements. The environment-specific requirements referred to in this section are those that help select IT product component requirements for profile inclusion.

This chapter describes the concerns that arise, and the steps that must be taken, in synthesizing profile functional and assurance components. It also illustrates the selection of these components by several examples. The chapter is divided into three sections. The first section describes several steps for synthesizing profile components. The second section addresses the notion of dependency analysis for profile components and component requirements. The third section contains a bibliography of useful literature references related to dependency analysis.

## 7.2 Synthesis of Profile Components

**Different Levels of Abstract Requirements.** The environment-specific requirements are used in selecting the functional and assurance components. These requirements can be stated at a level of abstraction that is higher than, lower than, or similar to that of the functional and assurance components. This variance in levels of abstraction exists because these requirements can be expressed in an unrestricted form. The requirements may be more abstract because they may reflect high-level security control objectives, organizational policies, regulations and directives. For example, environment-specific requirements may state that the computing facilities must reflect the separation of roles defined within an organization, or must reflect a document classification policy mandated by government directives. Similarly, the requirements may state that the control of access to documents processed within a computing facility must conform with a particular document processing policy (e.g., ORGCON).

Environment-specific requirements may be less abstract than those used in the functional and assurance components. Some may reflect the need to support a specific security standard or guideline (e.g., password guideline) while others may require a set of specific features and assurances deemed necessary in the environments of IT product use. For example, commercial security environments may require a specific set of: password complexity rules, location- and time-based access control rules, and security management rules. Other environments may mandate the use of a specific subject and object labeling policy, may require specific import or export policies for labeled objects, may mandate the use of specific forms of acceptance testing and test coverage, or may mandate a specific form of configuration management and trusted distribution.

Environment-specific requirements may have the same level of abstraction as that of the functional and assurance components because they may be derived from requirements of existing product standards. For example, some environment-specific requirements may be expressed by the requirements of the *Trusted Computer System Evaluation Criteria* (*TCSEC*) classes B2, B3, or A1, for high-assurance defense environments.

**Different Requirement Classifications**. The environment-specific requirements may be partitioned into components in a different manner than that used in the partitioning of the product generic requirements. Since the profile requirements ultimately drive the profile component selection, the different component partitionings must be resolved to ensure that the profile addresses all environment-specific requirements. The partitioning of generic product requirements into components and the rating of those components imply that, when interpreted at the product-requirement level, the environment-specific requirements must be expressed in terms of these components and their levels. For example, the environment-specific requirement class of "reliability of service" may contain specific requirements of limited service degradation, control of resource consumption, automated crash recovery based on checkpoint restart, and periodic back-up and restore operations. In terms of the product component requirements, the "reliability of service" requirement will be covered by the availability, trusted recovery, and security management components.

The partitioning of environment-specific requirements into product component requirements must take into account the rating of the component requirements because certain specific requirements may, in fact, be covered by individual requirements of multiple levels. For example, environment-specific requirements of access control may include all component level AC-2 (basic access control) and location-dependent authorization, which is a requirement included in component level AC-4 (fine-grain access control). Consequently, if component level AC-3 (extended access control) is selected, the environment-specific requirement would not be satisfied by the resulting profile, and if level AC-4 is selected, the resulting profile becomes overspecified because the requirements of AC-4 are unnecessarily included. The resolution of this problem is discussed in the next section.

The question of how the environment-specific requirements can be used to construct functional and assurance requirements for inclusion in a profile arises naturally, given the unconstrained level of abstraction in the environment-requirement definition. A key step in profile synthesis is that of selecting the functional and assurance components. The selection process is informal and, for this reason must be carefully justified in constructing and accepting a profile. When the level of the environment-specific requirements is close to that of the component requirements, two selection steps, *assignment* and *refinement*, are used.

### 7.2.1 Assignment

The assignment of environment-specific requirements to generic component requirements is performed when a component requirement corresponds to an environment-specific requirement. The correspondence is determined by analyzing the *intent* and *motivation* for both the environment-specific requirement and the product component requirement. A match of the motivation and intent for these requirements triggers the selection of the component requirement.

An assignment of environment-specific requirements to a component requirement also takes place when a component requirement is given a specific meaning. That is, a generic requirement of a component, which may require the definition of a rule, condition, or constant, becomes specific.

**Example 1: Assignment of specific constants**

In the identification and authentication component of the Commercial Security Protection Profile CS-2, the following **italicized** requirements assign specific default constants to successive unsuccessful login attempts and to the default of the required delay:

> The TCB shall end the attempted login session if the user performs the authentication procedure incorrectly for a number of successive times (i.e., a threshold) specified by an authorized system administrator. *The default threshold shall be three times*. When the threshold is exceeded, the TCB shall send an alarm message to the system console and/or to the administrator's terminal, log this event in the audit trail, and delay the next login by an interval of time specified by the authorized system administrator. *The default time interval shall be 60 seconds.* The TCB shall provide a protected mechanism to disable the user identity or account when the threshold of successive, unsuccessful login attempts is violated more than a number of times specified by the administrator. By default, this mechanism shall be disabled (as it may cause unauthorized denial of service).

Also, in the access control component of the Commercial Security Protection Profile CS-2, the following **italicized** requirement identifies a specific subject attribute (i.e., group identity) to which access rights are assigned:

> Object attributes shall include defined access rights (i.e**.,** read, write, execute) that can be assigned to subject attributes. *The TCB shall be able to assign object access rights to group identities.*

**Example 2: Assignment of specific authorization rules**

In the access control component of the Commercial Security Protection Profile CS-2, the following **italicized** requirement assigns specific authorization rules for subject references to objects:

> The TCB shall define and enforce authorization rules for the mediation of subject references to objects. These rules shall be based on the access control attributes of subjects and objects.
> *At a minimum, the authorization rules shall be defined as follows:*
>
> a. *The access rights associated with a user identifier shall take precedence over the access rights associated with any groups of which that user identifier is a member.*
> b. *When a user identifier can be an active member of multiple groups simultaneously, or if the access rights associated with the user identifier conflict with the access rights associated with any group in which the user is a member, it shall be possible for a system administrator to configure rules that combine the access rights to make a final access control decision.*
> c. *The TCB shall provide a protected mechanism to specify default access rights for user identifiers not otherwise specified either explicitly by a user identifier or implicitly by group membership.*

**Example 3: Assignment of specific conditions**

In the access control component of the Commercial Security Protection Profile CS-2, the following **italicized** requirement assigns specific conditions to the rule for assignment and modification of access control attributes for subjects and objects**.**

> The effect of these rules shall be that access rights to an object by users not already possessing access permission is assigned only by authorized users.
>
> *Only the current owner or system administrators can modify access control attributes of objects.*
>
> *There should be a distinct access right to modify the contents of an object's access control list (e.g., an "ownership" or "control" right).*

The component requirements are assigned a *null* environment-specific requirement whenever an environment-specific requirement is not assigned for a component. A null assignment implies that the component is not included in a profile (unless another component, which is required by another environment-specific requirement, depends upon it).

**Example 4: Null assignment**

In the Commercial Security Protection Profiles (CS-1, CS-2, CS-3), several assurance components were not selected for inclusion. The modular decomposition component, TCB structuring support, and TCB design disciplines were not selected because this profile does not require assurances about the internal TCB structure.

When an environment-specific requirement is assigned, it is possible that the component requirements used include some features that are not explicitly selected (i.e., an exact match is not possible). In this case, a *do not care* is assigned to the features and/or assurances not selected.

**Example 5: "Do not care" assignment**

In Example 2, the assignment of the specific authorization rules refers only to the selection of subject attributes for access authorization and does not include any specification of the object subset to which the authorization applies. This implies that a "do not care" is assigned to the generic requirement of identifying the authorization scope in the access control component. Similarly, a "do not care" assignment is implicitly made in Example 3. Although specific conditions are assigned to the rule for modification of access control attributes, a specific condition or rule was not assigned for attribute modification during object import and/or export operations.

### 7.2.2 Refinement

The refinement of a component requirement is necessary when the environment-specific requirements are less abstract (i.e., more specific) than the component requirements. As a consequence, one or more environment-specific requirements are added to a single component requirement. This represents a refinement of a component requirement. Note that the refinement of a component requirement differs from the assignment of environment-specific requirements to components. For example, a refinement of a component requirement may not assign any specific meaning to a requirement rule, condition, or constant. Instead, the refinement provides an elaboration of a generic component requirement in a specific environment.

**Example 6: Refinement of the trusted path component**

In the Commercial Security Protection Profile CS-2, the following **italicized** requirement refines the Trusted Path component TP-1 requirement:

> The TCB shall support a trusted communication path between itself and the user for initial identification and authentication. Communications via this path shall be initiated exclusively by a user.
> *The TCB shall provide a protected mechanism by which a data entry/display device may force a direct connection between the port to which it is connected and the authentication mechanism.*

**Example 7: Refinement of the authorization rules**

In the Commercial Security Protection Profile CS-2, the following **italicized** requirement refines the requirement for authorization rule definition and enforcement:

> The TCB shall define and enforce authorization rules for the mediation of subject references to objects. These rules shall be based on the access control attributes of subjects and objects.
> *For each object, the authorization rules of the TCB shall be based on a protected mechanism to specify a list of user identifiers or groups with their specific access rights to that object (i.e., an access control list).*

The assignment and refinement rules become necessary whenever the level of abstraction of the environment-specific requirements differs from that of the generic product components. However, when the partitioning or classification of environment specific requirements differs from that of the functional and assurance components, two additional selection steps, *decomposition* and *level-selection,* are used.

### 7.2.3 Decomposition

The decomposition of a specific requirement becomes necessary when that requirement must be assigned to multiple components of the generic product requirements during the interpretation process. Examples of decomposition are provided by both the specific requirements of the commercial domain illustrated in the NIST Minimum Security Functionality Requirements (MSFR) release 1.1 and by the specific requirements of labeled protection found in the TCSEC.

**Example 8: MSFR requirement decomposition into generic components**

*1. MSFR System Integrity Requirement -> Functional Components (AC, P, AD, SC, SM)*

| *Requirement* | *Component (paragraph)* |
|---|---|
| Separate process and address spaces | P-1 (1) |
| Verification of installed software using checksums & digital signatures | SC-3 |
| Restrict use of supervisory states | P-1 (1) |
| Audit use of operator consoles | AD-2 (2) |
| TCB software modification restricted to administrative users | SM-1 (4) |
| System maintenance limited to administrative users | SM-1 (4,5) |
| Validate correct operation of hardware & firmware elements | SC-1 |

*2. Data Integrity Requirement -> Functional Components (AC, SC, SM, ESU)*

| *Requirement* | *Component (paragraph)* |
|---|---|
| Record date & time object last modified | AC-4 (3) |
| Check file system and storage medium integrity | SC-3 |
| Display of system parameters and flags | SM-1 (2,3) |
| Directory/path search order | ESU-2 (1) |

*3. Reliability of Service -> Function Components ((AR, AF), TR, SM)*

| *Requirement* | *Component (paragraph)* |
|---|---|
| Degraded service operation | AF (TDB) |
| Controlled consumption of disk space, CPU usage | AR-1 |
| Recovery after system failure | TR-1 |
| Data backup & restore | SM-1 (4) |
| Checkpoint restarts | TR-4 (2) |

**Example 9: Decomposition of labeled component requirements into generic components**

*1. TCSEC Device Labeling Requirement (B2) -> Functional Components (AC, SE)*

| *Requirement* | *Component* |
|---|---|
| The TCB shall support the assignment of minimum and maximum security levels to all attached physical devices. | AC-2 (2), |
| These security levels shall be used by the TCB to enforce constraints imposed by the physical environments in which the devices are located. | I&A-2 (2) |

### 7.2.4 Level-Selection

The rating of functional and assurance components requires that specific component levels be selected when the environment-specific requirements are interpreted at the product level. However, an environment-specific requirement may exceed the requirements of a single level and may include individual requirements of higher levels. Whenever this happens, the selection of the component level follows a “low water mark” rule. That is, the selected level is the highest complete level required, but is augmented by individual requirements of higher levels. This leads to the development of new components from existing requirements, and ensures that the rating criteria used for the component levels does not impair flexibility in profile construction. Provided that an environment-specific requirement leads to the selection of at least one complete level (e.g., the low-water mark), different individual requirements of a higher level of the same component can be selected to augment the selected low-water-mark level.

**Example 10: Low-water-mark selection of component levels for MSFR requirements**

Access control requirements of the Commercial Security Protection Profiles CS-2 and CS-3 were derived from the specific requirements of the MSFR by using low-water-mark selection of levels.

> ***CS-2: AC-2+:*** These rules shall, either by explicit user action or by default, provide that objects are protected from unauthorized access.
>
> These rules shall allow authorized users to specify and control sharing of objects by named individuals or defined groups of individuals, or by both, and shall provide controls to limit propagation of access rights, ***(i.e., these rules shall define the distribution, revocation, and review of access control attributes). The controls defined by these rules shall be capable of specifying for each named object, a list of individuals and a list of groups of named individuals, with their respective access rights to that object. Furthermore, for each named object, it shall be possible to specify a list of named individuals and a list of groups of named individuals for which no access to the object is given [AC-4].*** These controls shall be capable of including or excluding access to the granularity of a single user.
>
> ***CS-3: AC-2+:*** If multiple access control policies are supported, the access control attributes corresponding to each individual policy shall be identified. The subject and object attributes shall accurately reflect the sensitivity and/or integrity of the subject or object. ***The subject's access control attributes also shall include time and location attributes that can be assigned to authenticated user identities [AC-4].***
>
> The TCB shall define and enforce authorization rules for the mediation of subject references to objects. These rules shall be based on the access control attributes of subjects and objects. These rules shall, either by explicit user action or by default, provide that objects are protected from unauthorized access. ***These rules shall include time-of-access and location-of-access controls defined for subjects and objects [AC-4].***

The rating of the functional and assurance components can also cause multiple levels of the same component to be selected when the environment-specific requirements are interpreted at the product level. Whenever this happens, the selection of the component level follows a “high water mark” rule. That is, the selected level is the maximum of all the levels separately selected from the same component.

**Example 11: High-water-mark selection of component levels for TCSEC requirements**

The system architecture requirements of the TCSEC class B3 include the following specific requirements:

*TCSEC System Architecture Requirement (B3) --> Modular Decomposition (MD)*

| *Requirement* | *Component* |
|---|---|
| The TCB shall be structured into well-defined modules | MD-2 |
| and | |
| Significant system engineering shall be directed towards minimizing the complexity of the TCB and excluding from the TCB modules that are not protection-critical | MD-3 |

The TCB structuring into modules requires the selection of assurance component MD-2. The minimization of the TCB complexity and the exclusion of protection-irrelevant modules from the TCB lead to the selection of the assurance component MD-3 because module exclusion requires the analysis of the correctness dependencies between modules. This is required to determine whether a protection-relevant module does not depend directly or indirectly on a module deemed to be protection-irrelevant and scheduled for removal from the TCB. Since the modular decomposition level MD-3 includes the requirements of level MD-2, level MD-3 is the high-water-mark level and thus it must be selected.

The system architecture and design specification and verification requirements of the TCSEC class A1 include the following specific requirements:

*TCSEC System Architecture Requirement (A1) - > Interface Definition (IF-2)*

*TCSEC Design Specification Requirement (A1) - > Interface Definition (IF-3)*

| *Requirement* | *Component* |
|---|---|
| The user interface shall be completely defined and all elements identified. | IF-2 |
| 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. | IF-3 |

Since the interface definition level IF-3 includes the requirements of level IF-2, level IF-3 is the high-water-mark level and thus, it must be selected.

Note that the decomposition and level-selection may require assignment and refinement and vice-versa. For example, the “low water mark” level selection, assignment, and refinement are illustrated by the requirements of access-control attribute administration in component AC-2+ of profiles CS-2 and CS-3.

## 7.3 Dependency Analysis

The analysis of the dependencies between functional and assurance components must be performed during profile construction. Such analysis helps (1) avoid inadequate, or incorrect, profile specification, (2) avoid overspecification of a profile, (3) determine the effect of profile changes (e.g., addition or removal of individual components or component requirements), and (4) analyze the compatibility of different protection profiles and harmonize different sets of component requirements (see Appendix E). This section illustrates and classifies functional and assurance dependencies. Examples are provided to show the use of dependency analysis in profile-compatibility analysis and profile-change analysis. This section is intended to enable protection profile developers to define consistent and coherent profiles that can be evaluated and used by independent organizations. It is further intended to motivate the analysis required when comparing different standards addressing information protection in IT products or when ensuring the preservation of previous investments (e.g., maintaining compatibility with the TCSEC).

### 7.3.1 Dependency Classification

Dependencies among the components of a product appear (1) among the functional components, (2) among assurance components, and (3) between the functional components and assurance components. Dependencies may also exist between the functional and assurance components and the product definition and operation. These dependencies help enlarge the application of a profile definition to widely-used products that might otherwise be considered inadequate for a specific protection profile. These dependencies can be analyzed in a similar manner as those of the first three classes, as this class does not introduce new dependency types. The role of classifying these dependencies is (1) to help achieve consistency and coherent profile definition, and (2) to decrease profile-definition susceptibility to inconsistent component classification of a component either as function or as an assurance.

Dependencies are classified into several types that are reminiscent of those that appear in the correctness analysis of large systems and products. This classification helps identify the important dependencies that are necessary to achieve the consistency and coherence of a protection profile.

### 7.3.2 Dependencies Among Functional Components

Dependencies among functional components arise because the functions that implement a component depend on functions implementing other components, or because different functions implementing different components must implement the same policy (properties) or requirement(s), individually or together. Thus,a distinction is made between the "uses" and "policy property" types of dependencies. There exists a "uses" dependency between two functional components, A and B, if the correctness of functions implementing A assumes the correctness of functions implementing B. There exists a "policy property" dependency between two functional components, A and B, if functions implementing both A and B must implement, either individually or together, a property or a condition required by the policy (e.g., the *-property, the simple security condition). Both "uses" and "policy property" dependencies may appear within a set of components, as shown in the balance of this section.

#### 7.3.2.1 "Uses" Dependency among Functional Components

"Uses" dependencies exist among different functional components of a TCB. Figure 7 illustrates "uses" dependencies among the different security policies supported by a TCB. These policies include access control, accountability (i.e., identification and authentication, system entry, trusted path, and audit), and availability. Figure 7 also illustrates "uses" dependencies among the security policies and the balance of the functional components (i.e., reference mediation, TCB logical protection, TCB least privilege operation, TCB ease-of -use, TCB start-up and recovery, TCB self-checking, and TCB physical protection). For example, a "uses" dependency arises among the access control and the TCB recovery components because access control can be correctly enforced only if the TCB recovery from failures and discontinuity of operations is correct.

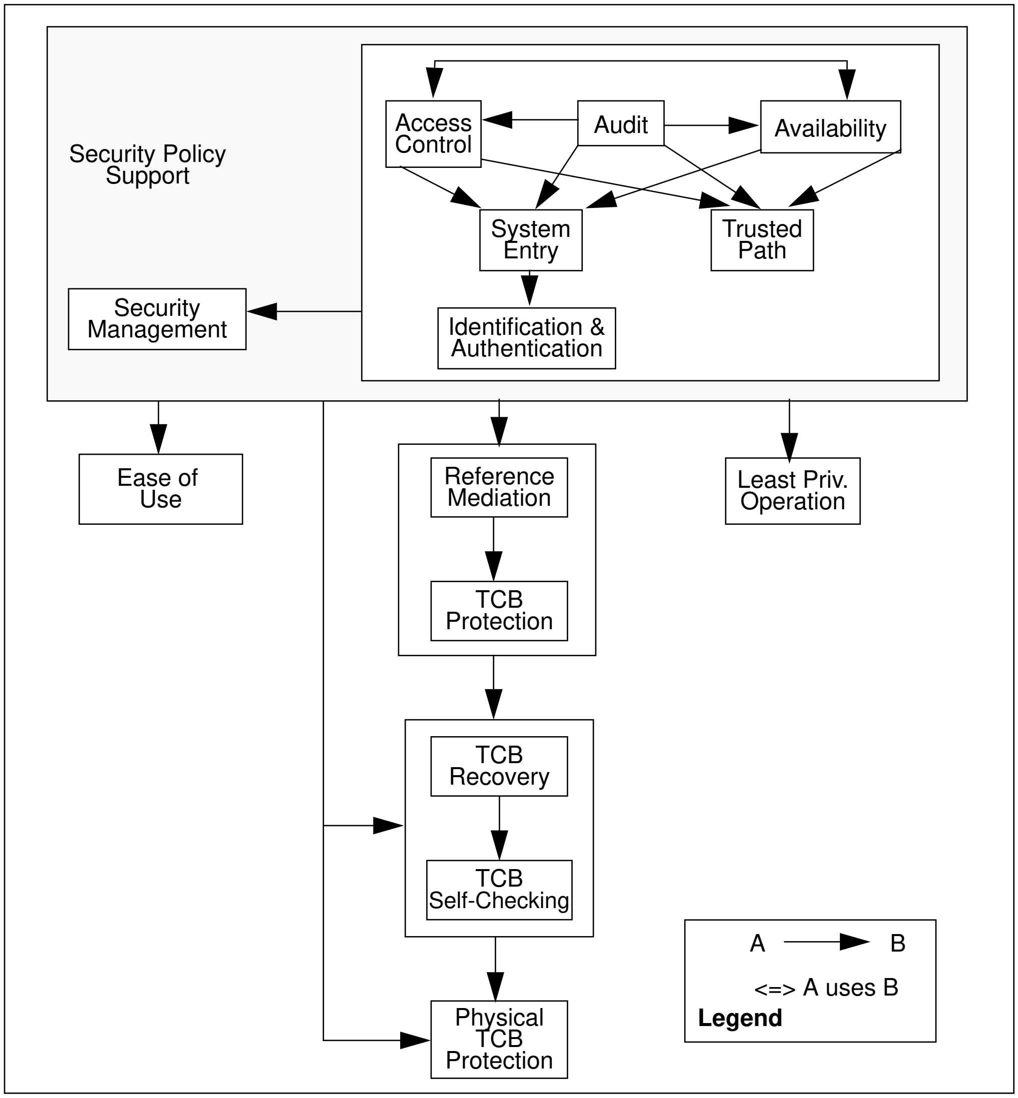



**Figure 7. Examples of Uses Dependencies among Functional Components.**

"Uses" dependencies also exist within a functional component of a TCB (i.e., among the individual requirements of a single component). Figure 8 illustrates several "uses" dependencies within the access control component of a TCB. For example, authorization has a "uses" dependency on attribute-administration because the access authorization functions are correct only if the distribution and revocation functions implementing attribute administration are correct (see Appendix C). A similar dependency appears within attribute administration (i.e., the access review function is correct only if the distribution and revocation functions are correct).

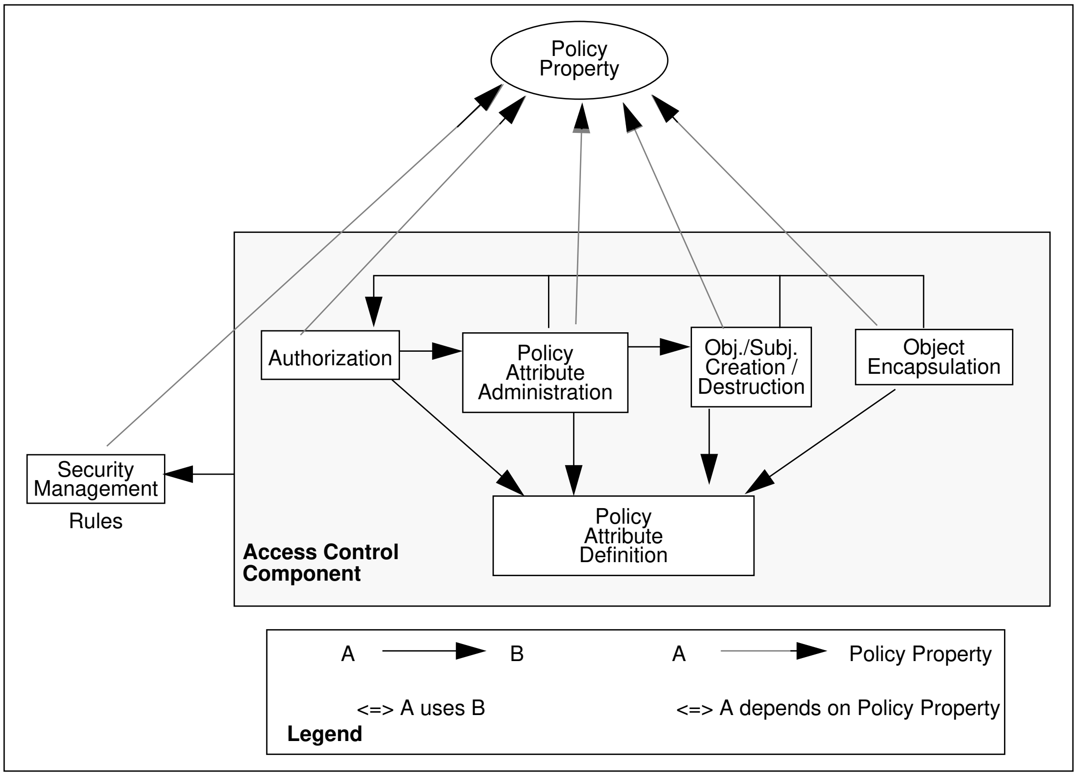



**Figure 8. Examples of Uses and Policy Properties Dependencies in Access Control.**

Note that both the "uses" dependency within a functional component and among functional components may cause *cyclic dependencies* to arise. A typical cyclic dependency is illustrated in Figure 9(a). Unprivileged subject references to objects can be mediated correctly only if TCB protection is provided, and TCB protection can be provided only if unprivileged subject references that attempt to modify objects implementing TCB isolation are denied by reference mediation. The removal of this cyclic dependency is illustrated in Figure 9(b). Removal is made possible by including a requirement (and corresponding function) for a specialized reference mediation that mediates only references to objects implementing TCB isolation.

Cyclic dependencies may arise among the requirements of several functional components, and individual requirements of functional components may be part of several cyclic dependencies. An example of multiple (i.e., three) cyclic dependencies is illustrated in Figure 9(c).

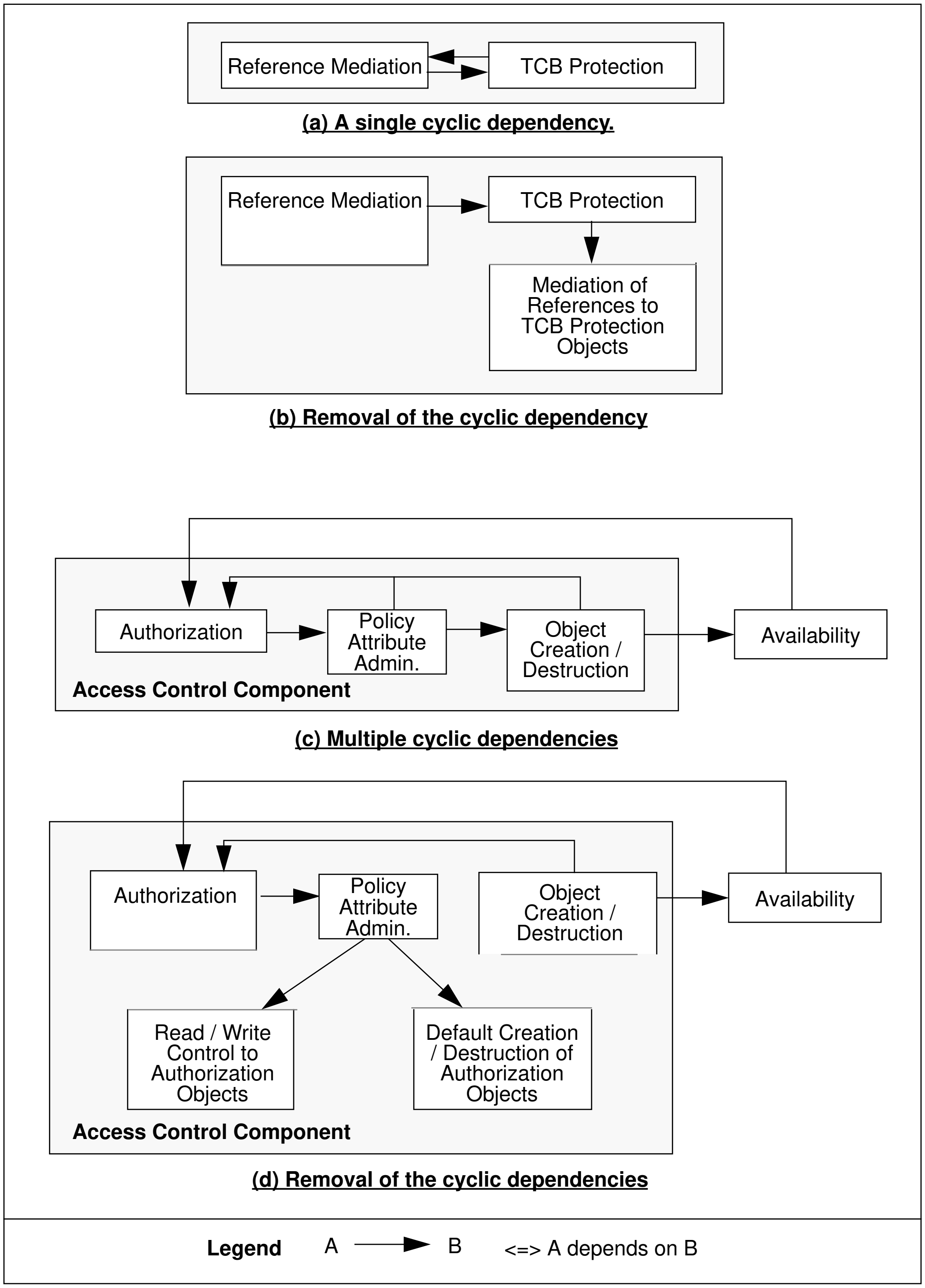



**Figure 9. Examples of Cyclic Dependencies and their Removal.**

In Figure 9(c), the first cyclic dependency is between access authorization and attribute administration. It arises not only because the authorization functions depend on attribute-administration functions (i.e., distribution and revocation functions), but also because the attribute-administration functions require authorization for reading and writing authorization objects (e.g., access control lists) to distribute, review, and revoke object access rights. The second cyclic dependency is between authorization and object creation and destruction. Object creation and destruction depends on authorization because, when objects are created (or destroyed) and placed in (or removed from) directories, the creation (or destruction) functions rely on access check functions that authorize directory modification. Attribute administration, however, depends on object creation and destruction because attribute-administration functions need to create authorization objects to specify object attributes (i.e., access rights). Hence, authorization depends, albeit indirectly, on object creation and destruction functions. The third cyclic dependency is between the availability component and the access control component. The availability function of modifying resource quotas can be correct only if the authorization function of access control is correct. Otherwise, arbitrary modifications of resource quotas may take place. Hence, availability depends on access authorization. Since the object creation component of access control depends on the resource allocation component of availability, a cyclic dependency arises because the authorization component depends indirectly on the object creation component.

Figure 9(d) illustrates the removal of the cyclic dependencies depicted in Figure 9(c). The cyclic dependency between authorization and attribute administration can be removed by including a requirement for a specialized authorization function that controls access only to authorization objects used for attribute administration. The cyclic dependency between attribute administration and object creation and destruction can be removed by including a requirement for default creation, initialization, and destruction of authorization objects for all other objects, within attribute administration. As illustrated in Figure 9(d), the removal of these two cyclic dependencies also causes the removal of the cyclic dependency between access authorization and the availability component of this example.

#### 7.3.2.2 Policy-Property Dependency

“Policy property” dependencies may be found within a single functional component and among different functional components of a TCB. Figure 8 also illustrates these policy property dependencies within a functional component of a TCB. For example, a property of an access control policy may be that “a subject may not view an object unless it has the read access right and may not alter an object unless it has the write access right for that object” (i.e., a property of access authorization which the TCB must implement). In an access control policy that supports this property, both the authorization and the attribute administration functions must maintain this property. Similarly, if the propagation of access rights to an object must be controlled, then a policy property may be that “unauthorized retention of access rights to an object cannot take place.” To satisfy this property, the access-right revocation function must be able to undo the effect of the access-right distribution function (i.e., a “policy property” dependency exists between the distribution and revocation functions of attribute administration). The two functions must have the same scope, granularity, and coverage (i.e., it must refer to the same set of subjects and objects, must refer to the same subject and object attributes, and must include or exclude the same conditions, such as transitivity).

Figure 10 illustrates several “policy property” dependencies among different functional components of a TCB. If components such as access control, audit, and availability are supposed to counter the same set of threats, then these components must satisfy the same policy properties, or requirements, either individually or together, and must have the same scope and granularity. For example, if the threat is that posed by malicious application programs (e.g., Trojan Horses in untrusted application programs), then the functional components of access control and availability policies (i.e., resource control) must be non-discretionary, and must control and audit the use of covert channels. These policies must also refer to the same set of subjects and objects (i.e., same scope) and to the same subject and object attributes (i.e., same granularity). Identification and authentication components must include non-discretionary attributes (e.g., confidentiality and/or integrity levels, roles) among the authorization data, and must control the users’ selection of these attributes during system entry. Trusted path support also becomes necessary.

### 7.3.2.3 Multiple Dependencies

A functional component may simultaneously depend on other functional components. A component may have (1) multiple “uses” dependencies, (2) multiple “policy-property” dependencies, or (3) combinations of ‘“uses” and “policy-property” dependencies. For example, Figure 7 shows that the access control, audit, and availability components have

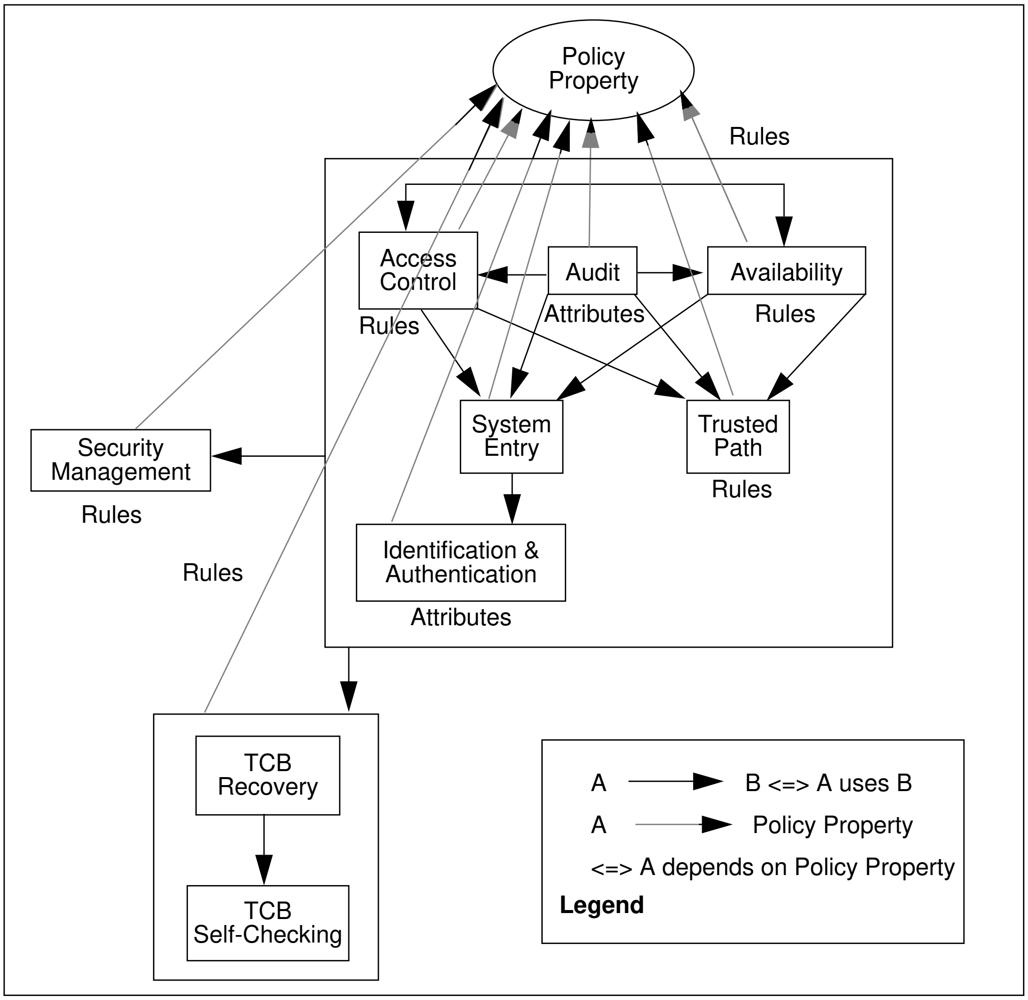



**Figure 10. Examples of Policy Property Dependencies.**

direct or indirect "uses" dependencies with all other functional components. Also, Figure 9 shows that object creation and destruction may have multiple direct "uses" dependencies (i.e., on authorization and availability).

Figures 8 and 10 suggest that, since multiple policies may be supported in a product, multiple policy properties will exist and, therefore, a component may have multiple "policy property" dependencies. The composition of policies within a product requires that multiple dependencies be analyzed to determine whether the composed policies satisfy the required system policy. For example, a profile may require that both a mandatory policy controlling information flow (via covert channels) and a discretionary policy be supported. The composition rules for the resulting TCB access control policy require that (1) both the mandatory and discretionary authorization rules be enforced on every subject and object protected by discretionary controls, and (2) the references issued by the enforcement modules of the discretionary policy be subject to the mediation specified by the mandatory rules. This precedence of enforcement is important whenever the exceptions returned by the enforcement of the two sets of rules are different. The reason is that if non-identical exceptions are returned by the two sets of rules, new covert channels may appear that would otherwise not appear had only the mandatory rules been enforced. These covert channels would violate the intent of the mandatory confidentiality policy. Similarly, the composition of distinct mandatory policies that individually control information flow may introduce additional flow violations that did not exist before composition. This suggests that the composition of policies within a profile introduces additional requirements for analyzing policy-property dependencies.

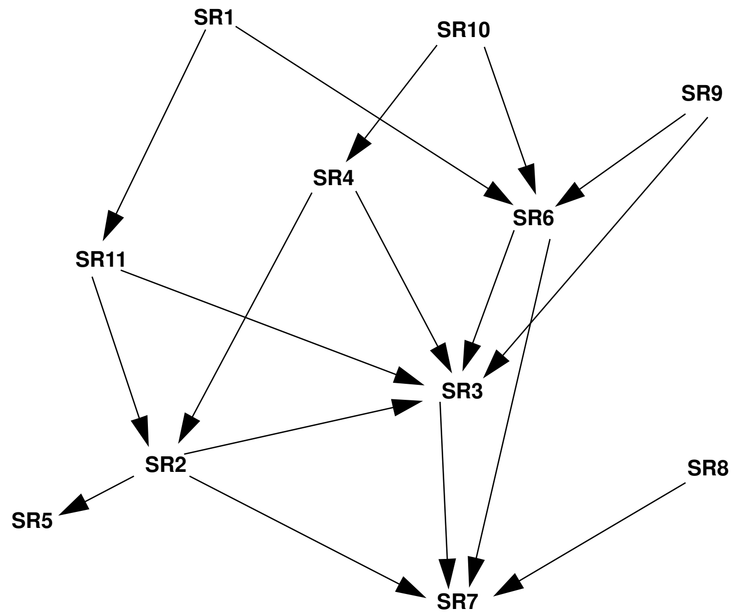
Figures 8 and 10 also illustrate that a component may have both "uses" and "policy-property" dependencies.

### 7.3.3 Dependencies Among Assurance Components

Dependencies arise among assurance components because some components use other components, or because different assurance components belong to the same assurance process. Thus, a distinction is made between "uses" dependencies and "assurance process" dependencies. A "uses" dependency exists between two assurance components, A and B, if obtaining assurance A requires that assurance B must be first obtained. An "assurance process" dependency exists between two assurance components, A and B, if both A and B represent two required stages of the same assurance process (e.g., development process, maintenance process in the development environment, and operation-support process).

#### 7.3.3.1 "Uses" Dependency among Assurance Components

The "uses" dependency can arise both among, and within, the components of the same assurance process and between the components of different assurance processes. Figure 11 illustrates several "uses" dependencies among, and within, the operational assurance requirements of the TCSEC class B2. For example, operational assurance SR1 depends on operational assurance SR6 because the TCB user (external) interfaces must be completely defined to establish the protection boundary of the TCB domain. SR1 also depends on the operational assurance SR11 (i.e., the reference validation mechanism) because the




LEGEND:

SR1 = The TCB shall maintain a domain for its own execution that protects it from external interference or tampering …

SR2 = The TCB shall maintain process isolation … the TCB shall … separate protection critical elements from those that are not.

SR3 = The TCB shall be structured into well-defined modules

SR4 = TCB modules shall be designed such that the principle of the least privilege be enforced

SR5 = Features in hardware, such as segmentation, shall be used to support logically distinct storage objects with separate attributes

SR6 = The user interface to the TCB shall be completely defined

SR7 = All elements of the TCB shall be identified

SR8 = … validate the correct operation of on-site hardware and firmware elements of the TCB

SR9 = … shall conduct a thorough search for covert storage channels and make a determination (…) of the maximum bandwidth of each identified channel (See the Covert Channels Guideline section)

SR10 = The TCB shall support separate operator and administrator functions

SR11 = The modules that contain the reference validation mechanism shall be identified

——► "Uses" dependency among B2 operational assurances

**Figure 11. Examples of Uses Dependencies Among the TCSEC B2 Operational Assurances.**

protection of the TCB domain can be established only if user references that attempt to modify TCB internal objects implementing TCB isolation are blocked by the reference validation mechanism. Operational assurance SR11 requires that the TCB be decomposed into modules. However, since the hardware/firmware modules that separate the protection-critical elements from those that are not protection-critical also contain reference validation checks, these modules must also be identified to satisfy operational assurance SR11. Hence, SR11 also depends on SR2. Also, operational assurance SR10 depends on operational assurance SR6 since the operator and administrator functions offer external TCB interfaces. SR10 depends on operational assurance SR4 because the operator and administrator functions are part of the TCB and, thus, must operate with the least privileges to accomplish their role Furthermore, the separation of operator and administrator functions implies that the operator and administrator must have special privileges representing different role authorities to invoke these functions. Similar reasoning applies to the other dependencies shown in Figure 11.

"Uses" dependencies appear between the components of the same assurance process because of the *types of specifications* and the *types of correspondences* between specifications used in the process. For example, both penetration-flaw and covert-channel identification methods depend on the types of TCB specification used. Specification-to-code correspondence depends on whether TCB design specifications are required and on the specific type of TCB design specifications (DIS or FIS). Generation of functional test conditions depends on policy-model interpretation in, or correspondence to, the TCB design specification, and test coverage using data-flow and path analysis depends on specification-to-code correspondence.

The "uses" dependency may arise between components of different assurance processes. For example, operational support components, such as flaw-discovery, tracking and repair, and also protection maintenance, TCB generation, and TCB distribution, depend on the configuration management component of the development environment. Naturally, the development evidence components depend on the components of the development process.

#### 7.3.3.2 Assurance-Process Dependencies

In contrast to the "uses" dependencies, the "assurance process" dependencies arise only among the stages of the same assurance process. For example, the operation-support process would be incomplete if only flaw discovery, but not tracking and repair, were performed. The maintenance process of the development environment would be meaningless if the configuration management component is implemented, but not the life-cycle component. If the procedures for controlling access to the configuration management systems are unspecified, the use of that IT product may become meaningless in some environments. Similarly, assurance of correct implementation of the TCB properties would not be available without the provision of a detailed design, architectural design, or TCB property definition.

Assurance-process dependencies help determine the assurance components necessary in an IT product and the chain of evidence that the product is correctly implemented. For example, the development assurance process may include the following design specification and verification requirements: (1) definition of the model for the access control policy, (2) TCB interface specification, (3) TCB implementation (e.g., source code), (4) valid interpretation of the model in the TCB (i.e., demonstration of consistency between the model and the TCB), and (5) TCB specification-to-code correspondence (i.e., demonstration of consistency between the TCB design specification and TCB source code). These requirements are process-dependent. Without any one of these requirements, the design specification and verification would be incomplete and the protection profile could become inadequate for the chosen environment of product use (e.g., it may not be possible to demonstrate the correct implementation of the reference monitor concept).

**Example 12: Missing process dependencies for the design specification and verification process**

Assurance requirements (1) - (5) listed above are found among those of the TCSEC class A1. The assurance requirements of class B3 lack the last assurance requirement, namely TCB specification-to-code correspondence (i.e., demonstration of consistency between the TCB design specification and TCB source code) and thus, the B3 design specification and verification process is incomplete. Note that since the complete analysis and testing of the reference validation mechanism is a requirement of the reference monitor concept, and since the assurance requirements of TCSEC class B3 require the demonstration of a reference monitor implementation, it is concluded that the class B3 assurance requirements do not completely satisfy the requirements of the reference monitor concept. (Although the other TCSEC classes lack this and other requirements of the design specification and verification process, their assurances are affected to a smaller degree because most of these classes do not include a requirement for demonstrable reference monitor implementation.)

### 7.3.4 Dependencies between Functions and Assurances

The analysis of the dependencies between functional and assurance components helps determine whether the selection of assurances made in the definition of a profile is consistent with the specific selection of functional-component requirements. That is, by definition, a functional component requirement has a "uses" dependency on an assurance requirement if the assurance requirement becomes necessary whenever the functional component requirement is used in a profile definition. In other words, the analysis of the dependencies between functional and assurance components helps determine whether a functional component can be correctly designed, analyzed, implemented, and evaluated given the selected set of assurance components.

Note that, based on the definition of a "uses" dependency and on the definition and classification of functions and assurances used in this standard, obtaining an assurance should be independent of the presence of any protection function; i.e., obtaining and demonstrating an assurance for a protection function should not require that other protection functions be added to a TCB. This assurance independence of functional components is also justified by the observation that assurances contribute only to the elimination of internal TCB design and implementation errors but do not counter any threat posed by external users or untrusted applications.

The dependencies between functional and assurance components are "uses" dependencies. These dependencies are illustrated by the following examples:

a. Whenever functions of distinct security policies are supported (e.g., composed) within the TCB, the TCB interface must be designed so that it is consistent with the properties of the overall TCB security policy. By the definition of dependency between functional and assurance components, the access control functions depend on the TCB interface design.

b. Whenever mandatory confidentiality or integrity policies are supported within a TCB to establish information flow boundaries among untrusted applications, a covert-channel analysis must be performed. Thus, the access control policies used depend on covert-channel analysis.

c. Whenever different identification and authentication policies are used within a TCB (e.g., user-chosen passwords or one-time passwords generated by password devices), the selection of test condition and test coverage types is based on the properties of those policies. Password length, lifetime, and complexity testing is performed for policies that allow users to choose their own passwords, whereas only the analysis of the complexity of one-time passwords is performed for policies using one-time passwords (since these passwords have fixed length and lifetime).

d. Whenever the reference validation mechanism is implemented within a TCB, the access control policies defined have a dependency on the type of specification-to-code correspondence method. For example, the correspondence methods used to show that discretionary access control requirements are implemented by source code may be based on establishing the correspondence between state transitions of a policy model and those of the source code. These methods differ from those based on information flow and non-interference, which are used to show that the source code does not introduce information flows to those flows found in the interface specifications.

#### 7.3.4.1 Relationship to other Function and Assurance Classifications

It is important to note that other, equally valid, classifications of functional and assurance components, which differ from the one defined in this standard, may cause assurances to depend on access control components. For example, TCB recovery, covert channel handling, trusted facility management, and the TCB privileged (i.e., least privilege) operations may be considered to be *operational* assurances (see the TCSEC). As shown in Figures 8 and 10, some operational assurances become policy-property dependent on the access control components because some of these assurances can only be obtained if the policy properties are defined. Cyclic dependencies may also arise between these components; e.g., between trusted recovery assurance and access control.

The specific classification of TCB functional and assurance components used in this standard does not affect the dependencies among the profile components. For example, the dependencies among operational assurances of the TCSEC B2 class products are described as (1) “uses” dependencies among assurance components of the development process, (2) “uses” dependencies among functional components, and (3) “uses” dependencies between functional components on assurance components of this standard. This is illustrated by the examples of the next section. It is important to note that regardless of how the functional and assurance components are classified, the existence of dependencies identified among those components does not change. In this sense, dependency analysis removes the susceptibility of a profile definition and analysis to different classifications of functional and assurance components.

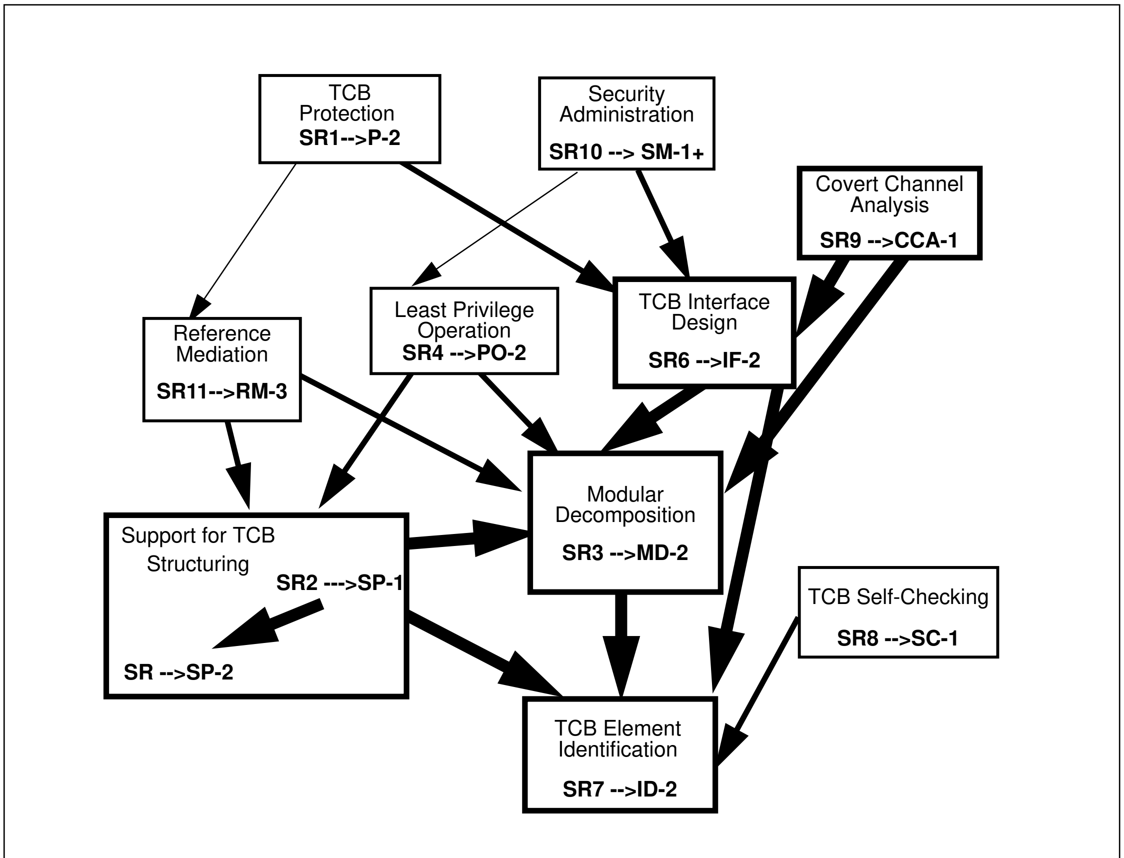
### 7.3.5 Examples of Using Dependency Analysis

The use of dependency analysis is illustrated by two examples. First, functional and assurance components are selected for a protection profile that is intended to include the B2 operational assurances of the TCSEC (see protection profile LP-2). Second, dependency analysis is used in profile enhancement. The example illustrates the role of dependency analysis when the B2 assurances are enhanced by the B3 assurances (see protection profile LP-3).

**Example 13: Analysis of profile compatibility**

The result of decomposing the TCSEC B2 operational assurance requirements into the functional and assurance components of this standard is illustrated in Figure 12. After decomposing the B2 requirements, it must be established that the decomposition does preserve the dependencies (e.g., the "uses" dependencies) that exist among the B2 operational assurances. To establish that the dependencies are preserved, the assignment and level-selection steps must also be performed. Figure 12 shows the assignment and level-selection performed for the decomposed B2 assurance requirements. With the exception of specific requirements, SR1, SR8, SR10 and SR11, which are classified as functional component requirements by this standard, all other specific B2 operational assurance requirements (see Figure 11) correspond to assurance components of this standard.

Figure 12 illustrates the fact that reclassification of an assurance component as a functional component does not affect the existing dependencies. This figure shows that the TCB interface design (IF-2) relies on the decomposition of the TCB into modules and the identification of the modules that offer external TCB interfaces (MD-2). TCB modular decomposition cannot be performed without the identification of the TCB elements (ID-2). Storage channel analysis (CCA-1) needs both the TCB interface design (IF-2) and the modular decomposition of the TCB (MD-2); the former is needed for defining the covert-storage channels in terms of TCB system calls and parameters, whereas the latter is needed for source-code level identification of information flows. Support for TCB structuring (SP-2) can be effective only if both the modular decomposition of the TCB (MD-2) and the




LEGEND:

SR1 = The TCB shall maintain a domain for its own execution that protects it from external interference or tampering …

SR2 = The TCB shall maintain process isolation … the TCB shall … separate protection critical elements from those that are not.

SR3 = The TCB shall be structured into well-defined modules

SR4 = TCB modules shall be designed such that the principle of the least privilege be enforced

SR5 = Features in hardware, such as segmentation, shall be used to support logically distinct storage objects with separate attributes

SR6 = The user interface to the TCB shall be completely defined

SR7 = All elements of the TCB shall be identified

SR8 = … validate the correct operation of on-site hardware and firmware elements of the TCB

SR9 = … shall conduct a thorough search for covert storage channels and make a determination (…) of the maximum bandwidth of each identified channel (See the Covert Channels Guideline section)

SR10 = The TCB shall support separate operator and administrator functions

SR11 = The modules that contain the reference validation mechanism shall be identified

“Uses” dependency among functional components

“Uses” dependency between functional and assurance components

“Uses” dependency among assurance components.

**Figure 12. Examples of Uses Dependencies Among Components Corresponding to B2 Operational Assurances.**

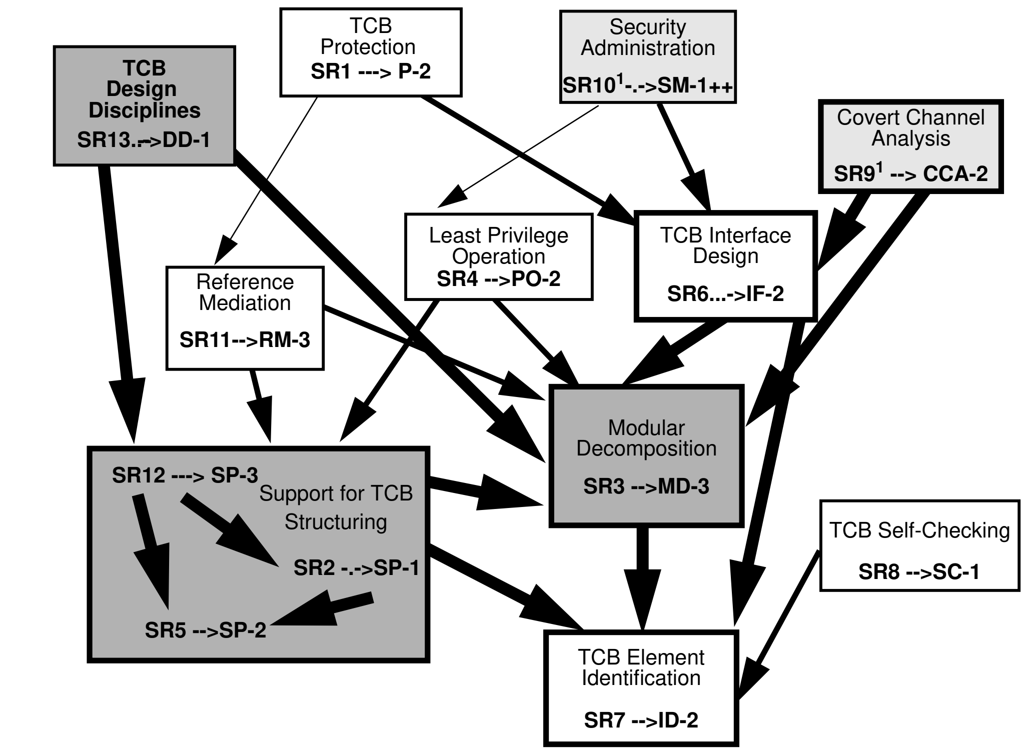
identification of the TCB elements (ID-2) are available. The isolation of TCB processes and the separation of the protection critical TCB elements from the non-critical ones (SP-2) requires the modular decomposition of the TCB elements. Modular decomposition and separation can only be done after the TCB elements are identified and justified (ID-2). Note, however, that the dependency of the specific requirement SR2 on SR5 illustrated in Figure 11 does not correspond to an inter-component dependency in Figure 12. Instead, it corresponds to the implicit dependency between the component levels SP- 2 and SP-1; i.e., SP-1 is included in SP-2. The high-water-mark level selection implies that only level SP-2 is selected for profile inclusion.

Similar reasoning can be used to show that the rest of the dependencies among the B2 operational assurances are preserved by the decomposition, assignment, and level-selection steps leading to the functional and assurances components synthesized in Figure 12 (and in the protection profile LP-2).

**Example 14: Enhancing profile requirements**

Enhancing the component requirements of a protection profile (1) can introduce new dependencies and (2) lead to new level selections in profile synthesis. For example, enhancing the operational assurance requirements of the TCSEC B2 class to obtain those of the TCSEC B3 class introduces both new dependencies and level selections. Figure 13 illustrates the new level selections for the corresponding profile components. For example, the B3 requirement SR10', which replaces the B2 requirement SR10, implies that component SM-1++ must replace component SM-1+ in the corresponding profile (see protection profile LP-3). Furthermore, the B3 covert-channel analysis requirement SR9', which replaces the B2 storage-channel analysis requirement SR9, implies that the component CCA-2 must replace component CCA-1 in the corresponding profile (see protection profile LP-3).

Figure 13 also illustrates the new dependencies introduced by the transition from operational assurances of class B2 to those of class B3 in the TCSEC. New dependencies appear between requirements SR13 and SR3, between requirements SR13 and SR2, between requirements SR12 and SR2, and between requirements SR12 and SR5. These new dependencies cause the high-water-mark selection of levels MD-3 and SP-3. Within the development process, the minimization of the TCB complexity (as required by SR13) depends on the modular decomposition of the TCB (as required by SR3), and on the analysis of the "uses" dependencies among modules. If a module containing a protection-relevant function also depends upon the correctness of another module, then that other




LEGEND:

SR1 = The TCB shall maintain a domain for its own execution that protects it from external interference or tampering …

SR2 = The TCB shall maintain process isolation … the TCB shall … separate protection critical elements from those that are not.

SR3 = The TCB shall be structured into well-defined modules

SR4 = TCB modules shall be designed such that the principle of the least privilege be enforced

SR5 = Features in hardware, such as segmentation, shall be used to support logically distinct storage objects with separate attributes

SR6 = The user interface to the TCB shall be completely defined

SR7 = All elements of the TCB shall be identified

SR8 = … validate the correct operation of on-site hardware and firmware elements of the TCB

SR9[1] = … shall conduct a thorough search for covert channels and to make a determination (…) of the maximum bandwidth of each identified channel

SR10[1] =… Non-security functions that can be performed in security admin. role shall be limited strictly to those essential to performing the security role effectively

SR11 = The modules that contain the reference validation mechanism shall be identified

SR12 = The TCB shall be designed and structured to use a conceptually simple prot. mechanism ... This mechanism shall play a central role in ... internal structuring of the TCB and the System

SR13 = ...significant use of layering, abstraction and data hiding. ...minimizing the complexity of the TCB and excluding from the TCB modules that are not protection-critical

“Uses” dependency among functional components

“Uses” dependency between functional and assurance components

“Uses” dependency among assurance components.

**Figure 13. Example of Uses Dependencies among the Function and Assurance Components Corresponding to the B3 Operational Assurances.**

module cannot be removed from the TCB. Since the modular decomposition level MD-3, not MD-2, includes the analysis of the inter-module correctness relationships, level MD-3 must be selected. Similarly, the run-time enforcement of data hiding, abstraction, and layering must be supported by a TCB protection mechanism with precisely defined semantics (e.g., rings or domains of protection). Otherwise, it cannot be reasoned that the TCB structuring is correctly enforced. This suggests that the TCB structuring support component SP-3, not SP-2, must be selected for profile inclusion.

The dependency of the specific requirement SR12 on SR2 and SR5 illustrated in Figure 13 does not correspond to an inter-component dependency. Instead, it corresponds to the implicit dependency between the component levels SP-3, SP- 2 and SP-1; i.e., SP-1 is included in SP-2, and SP-2 is included in SP-3. Note that, even if component level SP-2 is required by other component levels (e.g., by RM-3 in Figure 12), the high-water-mark level selection implies that component level SP-3 must be selected.

## 7.4 Bibliographic Notes

TBD.

# GLOSSARY

**Access** - Ability and means to communicate with (i.e., input to or receive output from), or otherwise make use of any information, resource, or object in an Information Technology (IT) Product. Frequently used as a verb, contrary to the rules of grammar.

Note: An individual does not have “access” if the proper authority or a physical, technical, or procedural measure prevents them from obtaining knowledge or having an opportunity to alter information, material, resources, or components.

**Access Control** - Process of limiting access to the resources of an IT product only to authorized users, programs, processes, systems, or other IT products.

**Access Control List** -A list of subjects that are authorized to have access to some object(s). Usually, this list contains entries consisting of identifiers of users and groups of users and access rights.

**Access Control Mechanism** - Security safeguards designed to detect and prevent unauthorized access, and to permit authorized access in an IT product.

**Access Mediation** - Process of monitoring and controlling access to the resources of an IT product, including but not limited to the monitoring and updating of policy attributes during accesses as well as the protection of unauthorized or inappropriate accesses (see Access Control).

**Accountability** - Means of linking individuals to their interactions with an IT product, thereby supporting identification of and recovery from unexpected or unavoidable failures of the control objectives.

**Accreditation -** Formal declaration by a designated approving authority that an Automated Information System (AIS) is approved to operate in a particular security configuration using a prescribed set of safeguards.

**Application Program Interface** - System access point or library function that has a well-defined syntax and is accessible from application programs or user code to provide well-defined functionality.

**Assignment** - Requirement in a protection profile taken directly as stated, without change, from the list of components or derived by placing a bound on a threshold definition.

Note: The assignment of environment-specific requirements to generic component requirements is performed when a component requirement corresponds to an environment-specific requirement.

**Assurance** - (See Profile Assurance and IT Product Assurance).

**Audit** - Independent review and examination of records and activities to determine compliance with established usage policies and to detect possible inadequacies in product technical security policies of their enforcement.

**Audit Trail** - Chronological record of system activities to enable the reconstruction and examination of the sequence of events and/or changes in an event. [NSTISSI 4009]

Note: Audit trail may apply to information in an IT product or an AIS or to the transfer of COMSEC material.

**Authenticate** - Verify the identity of a user, user device, or other entity, or the integrity of data stored, transmitted, or otherwise exposed to unauthorized modification in an IT product.

**Authentication** - Means of verifying an entity's (e.g., individual user, machine, software component) eligibility to receive specific categories of information.

**Authorization** - Access rights granted to a user, program, or process. [NSTISSI 4009]

**Authorized** - Entitled to a specific mode of access.

**Automated Information System (AIS)** - Any equipment or interconnected systems or subsystems of equipment that is used in the automatic acquisition, storage, manipulation, management, movement, control, display, switching, interchange, transmission or reception of data and includes computer software, firmware, and hardware. [NSTISSI 4009]

Note: Included are computers, word processing systems, networks, or other electronic information handling systems, and associated equipment.

**Availability** - Ability to access a specific resource within a specific time frame as defined within the IT product specification.

**Bandwidth** - Rate at which information is transmitted through a channel. (See channel capacity)

Note: Bandwidth is originally a term used in analog communication, measured in Hertz, and related to information rate by the "sampling theorem" (generally attributed to H. Nyquist although the theorem was in fact known before Nyquist used it in communication theory). Nyquist's sampling theorem says that the information rate in bits (samples) per second is at most twice the bandwidth in Hertz of an analog signal created from a square wave. In a covert-channel context "bandwidth" is given in bits/second rather than Hertz and is commonly used, in an abuse of terminology, as a synonym for information rate.

**Bell-La Padula Security Model** - Any formal state-transition model of a technical security policy for an AIS that presents (a) Access Constraints (including initial-state constraints and variants or the simple security and star properties), (b) allowed state transitions (called "rules of operation"), and (c) a proof that the allowed state transitions guarantee satisfaction of the constraints.

**Category** - Restrictive label that has been applied to both classified and unclassified data, thereby increasing the requirement for protection of, and restricting the access to, the data. [NSTISSI 4009]

Note: Examples include sensitive compartmented information and proprietary information. Individuals are granted access to special category information only after being granted formal access authorization.

**Certification** - Comprehensive evaluation of the technical and nontechnical security features of an AIS and other safeguards, made in support of the accreditation process, to establish the extent to which a particular design and implementation meets a set of specified security requirements. [NSTISSI 4009]

**Channel Capacity** - Maximum possible error-free rate, measured in bits per second, at which information can be sent along a communications path.

**Clear-text** - Intelligible data, the semantic content of which is available. [ISO]

**Confidentiality** - Assurance that information is not disclosed to inappropriate entities or processes.

**Configuration** - Selection of one of the sets of possible combinations of features of a system. [ITSEC]

**Consumers** - Individuals or groups responsible for specifying requirements for IT product security (e.g., policy makers and regulatory officials, system architects, integrators, acquisition managers, product purchasers, and end users.

**Control Objective** - Required result of protecting information within an IT product and its immediate environment.

**Countermeasure** - Action, device, procedure, technique, or other measure that reduces the vulnerability of an AIS. [NSTISSI 4009]

**Covert Channel** - Unintended and/or unauthorized communications path that can be used to transfer information in a manner that violates an AIS security policy. (See overt channel and exploitable channel.) [NSTISSI 4009]

**Covert Storage Channel** - Covert channel that involves the direct or indirect writing to a storage location by one process and the direct or indirect reading of the storage location by another process. [NSTISSI 4009]

Note: Covert storage channels typically involve a finite resource (e.g., sectors on a disk) that is shared by two subjects at different security levels.

**Covert Timing Channel** - Covert channel in which one process signals information to another process by modulating its own use of system resources (e.g., central processing unit time) in such a way that this manipulation affects the real response time observed by the second process. [NSTISSI 4009]

**Decomposition** - Requirement in a protection profile that spans several components.

Note: The decomposition of a specific requirement becomes necessary when that requirement must be assigned to multiple components of the generic product requirements during the interpretation process.

**Definition** - an informal statement expressing the essential characteristics of one or more facts.

**Demonstration** - an act or process of producing conclusive evidence for one or more facts. (A demonstration is more rigorous than an explanation and less rigorous than a proof).

**Dependency** - Condition in which the correctness of one or more functions (or assurances) is contingent (depends for its correctness) on the correctness of another function(s) (or assurances). Notion also used in describing the relationships among TCB subsets [NCSC-TG-021]. A TCB subset ***A*** depends for its correctness on TCB subset ***B*** if and only if the (engineering) arguments of the correct implementation of ***A*** with respect to its specification assume, wholly or in part, that the specification of ***B*** has been implemented correctly.

**Description** - an enumeration of facts and their characteristics.

**Designated Approving Authority (DAA)** - Official with the authority to formally assume responsibility for operating an IT product, an AIS, or network at an acceptable level of risk.

**Development Assurance** - Sources of IT product assurance ranging from how a product was designed and implemented to how it is tested, operated and maintained.

**Development Assurance Component** - Fundamental building block, specifying how an IT product is developed, from which development assurance requirements are assembled.

**Development Assurance Package** - Grouping of development assurance components assembled to ease specification and common understanding of how an IT product is developed.

**Development Assurance Requirements** - Requirements in a protection profile which address how each conforming IT product is developed including the production of appropriate supporting developmental process evidence and how that product will be maintained.

**Discretionary Access Control** - Methods of restricting access to objects or other resources based primarily on the instructions of arbitrary unprivileged users.

**Domain** - Unique context (e.g., access control parameters) in which a program is operating.

Note: A subject's domain determines which access-control attributes an object must have for a subject operating in that domain to have a designated form of access.

**Encapsulated Object** -A data structure whose existence is known, but whose internal organization is not accessible, except by invoking the encapsulated subsystem that manages it.

**Encapsulated Subsystem** -A collection of procedures and data objects that is protected in a domain of its own so that the internal structure of a data object is accessible only to the procedures of the encapsulated subsystem that the procedures may be called only at designated domain entry points. Encapsulated subsystem, protected subsystem, and protected mechanisms of the TCB are terms that may be used interchangeably.

**Environment** - All entities (users, procedures, conditions, objects, AISs, other IT products) that interact with (affect the development, operation and maintenance of) that IT product.

**Evaluation** - Technical assessment of a component's, product's, subsystem's, or system's security properties that establishes whether or not the component, product, subsystem, or system meets a specific set of requirements.

Note: Evaluation is a term that causes much confusion in the security community, because it is used in many different ways. It is sometimes used in the general English sense (judgement or determination of worth or quality). Based on common usage of the term in the security community, one can distinguish between two types of evaluation: (1) evaluations that exclude the environment, and (2) evaluations that include the environment. This second type of evaluation, an assessment of a system's security properties with respect to a specific operational mission, is termed *certification* within this document. Evaluations that exclude the environment, the type of evaluations considered herein, are assessments of the security properties against a defined criteria.

**Evaluation Assurance** - Source of IT product assurance based on the kind and intensity of the evaluation analysis performed on the product.

**Evaluation Assurance Component** - Fundamental building block, specifying the type and the rigor of required evaluation activities, from which evaluation assurance requirements are assembled.

**Evaluation Assurance Package** - Grouping of evaluation assurance components assembled to ease specification and common understanding of the type and the rigor of required evaluation activities.

**Evaluation Assurance Requirements** - Requirements in a protection profile which address both the type and the rigor of activities that must occur during product evaluation.

**Evaluators** - Individuals or groups responsible for the independent assessment of IT product security (e.g., product evaluators, system security officers, system certifiers, and system accreditors).

**Explanation** - a description and its justification; an enumeration of facts, their characteristics, and their cause or reason. (An explanation is less rigorous than both a demonstration and a proof.)

**Exploitable Channel** - Covert channel that is usable or detectable by subjects external to the AIS's trusted computing base and can be used to violate the AIS's technical security policy. (See covert channel.)

**External Security Controls** - Measures which include physical, personnel, procedural, and administrative security requirements and a separate certification and accreditation process that govern physical access to an IT product.

Note: These measures constitute assumptions and boundary conditions that are part of the environment described in a protection profile.

**Flaw** - Error of commission, omission, or oversight in an IT product that may allow protection mechanisms to be bypassed.

**Formal Security Policy Model** - Mathematically precise statement consisting of (a) a formal technical security policy (given by constraints on a Product's external interface and/or constraints on the handling of controlled entities internal to the Product), (b) rules of operation that show how the definition of security is to be enforced, and (c) a formal proof showing that the rules of operation guarantee satisfaction of the definition of security. [NCSC-TG-010]

**Formal Specification** - Statement about a product made using the restricted syntax and grammar of a formal reasoning system and a set of terms that have been precisely and uniquely defined of specified.

Note: The formal statement should be augmented by an informal explanation of the conventions used and the ideas being expressed. A well-formed syntax and semantics with complete specification of all constructs used must be referenced.

**Functional Component** - Fundamental building block, specifying what an IT product must be capable of doing, from which functional protection requirements are assembled.

**Functional Package** - Grouping of functional components assembled to ease specification and common understanding of what an IT product is capable of doing.

**Functional Protection Requirements** - Requirements in a protection profile which address what conforming IT products must be capable of doing.

**Functionality** - Set of functional protection requirements to be implemented in IT products.

**Generic Threat** - Class of threats with common characteristics pertaining to vulnerabilities, agents, event sequences, and resulting misfortunes.

**Granularity** - Relative fineness or coarseness to which an access control mechanism or other IT product aspect can be adjusted.

Note: Protection at the file level is considered course granularity, whereas protection at the field level is considered to be finer granularity.

**Granularity of a Requirement** - Determination of whether a requirement applies to all the attributes of users, subjects or objects, and all TCB functional components.

**Group** - Named collection of user identifiers.

**Identification** - Process that enables recognition of an entity by an IT product.

**Informal Specification** - Statement about (the properties of) a product made using the grammar, syntax, and common definitions of a natural language (e.g., English).

Note: While no notational restrictions apply, the informal specification is also required to provide defined meanings for terms which are used in a context other than that accepted by normal usage.

**Information Protection Policy** - Set of laws, rules, and practices that regulate how an IT product will, within specified limits, counter threats expected in the product's assumed operational environment.

**Internal Security Controls** - Mechanisms implemented in the hardware, firmware, and software of an IT product which provide protection for the IT product.

**Integrity** - Correctness and appropriateness of the content and/or source of a piece of information.

**Least Privilege** - Principle that requires that each subject be granted the most restrictive set of privileges needed for the performance of authorized tasks. [NSTISSI 4009]

Note: Application of this principle limits the damage that can result from accident, error, or unauthorized use of an AIS.

**Mandatory Access Control** - Means of restricting access to objects based on the sensitivity (as represented by a label) of the information contained in the objects and the formal authorization (i.e., clearance) of subjects to access information of such sensitivity. (See non-discretionary access control.)

**Mechanism** - Operating system entry point or separate operating system support program that performs a specific action or related group of actions.

**Need-to-know** - Access to, or knowledge or possession of, specific information required to carry out official duties. [NSTISSI 4009]

**Non-Discretionary Access Control** - Means of restricting access to objects based largely on administrative actions.

**Normal Operation** - Process of using a system. [ITSEC]

**Object** - Controlled entity that precisely gives or receives information in response to access attempts by another (active) entity.

Note: Access to an object implies access to the information contained in that object. Examples of objects include: subjects, records, blocks, pages, segments, files, directories, directory trees and programs, as well as bits, bytes, words, fields, processors, I/O devices, video displays, keyboards, clocks, printers, and network nodes.

**Object Encapsulation** - viz., encapsulated object.

**Organizational Security Policy** - Set of laws, rules, and practices that regulate how an organization manages, protects, and distributes sensitive information.

**Overt Channel** - Communications path within a computer system or network that is designed for the authorized transfer of data. (See covert channel) [NSTISSI 4009]

**Owner** - User granted privileges with respect to security attributes and privileges affecting specific subjects and objects.

**Password** - Protected/private character string used to authenticate an identity or to authorize access to data. [NSTISSI 4009]

**Penetration Testing** - Security testing in which evaluators attempt to circumvent the security features of an AIS based on their understanding of the system design and implementation. [NSTISSI 4009]

**Primitive** - Orderly relation between TCB subsets based on dependency. [NCSC-TG-021]

Note: A TCB subset ***B*** is more primitive than a second TCB subset ***A*** (and ***A*** is less primitive than ***B***) if ***A*** directly depends on ***B*** or a chain of TCB subsets from ***A*** to ***B*** exists such that each element of the chain directly depends on its successor in the chain.

**Privilege** - Special authorization that is granted to particular users to perform security relevant operations.

**Process** - A program in execution on a processor which represents a scheduling and accounting (and sometimes a concurrency and recovery) entity in a computer system.

**Producers** - Providers of IT product security (e.g., product vendors, product developers, security analysts, and value-added resellers).

**Product** - Package of IT software and/or hardware designed to perform a specific function either stand alone or once incorporated into an IT system.

**Product Rationale** - Overall justification; including anticipated threats, objectives for product functionality and assurance, technical security policy, and assumptions about the environments and uses of conforming products; for the protection profile and its resulting IT product.

**Profile** - Detailed security description of the physical structure, equipment component, location, relationships, and general operating environment of an IT product or AIS. (See Protection Profile.)

**Profile Assurance** - Measure of confidence in the technical soundness of a protection profile.

**Proprietary Information** - Information that is owned by a private enterprise and whose use and/or distribution is restricted by that enterprise.

Note: Proprietary information may be related to the company's products, business, or activities, including but not limited to: financial information, data or statements; trade secrets; product research and development information; existing and future product designs and performance specifications; marketing plans or techniques; schematics; client lists; computer programs; processes; and trade secrets or other company confidential information.

**Protected Mechanism** - See encapsulated subsystem.

**Protection Philosophy** - Informal description of the overall design of an IT product that shows how each of the supported control objectives is dealt with.

**Protection Profile** - Statement of security criteria; shared by IT product producers, consumers, and evaluators; built from functional, development assurance, and evaluation assurance requirements; to meet identified security needs through the development of conforming IT products.

**Protection Profile Family** - Two or more protection profiles with similar functional requirements and rationale sections but with different assurance requirements.

**Proof** - the process of establishing the validity of one or more statements; the process of establishing a the truth of a fact. (A proof is more rigorous than both a demonstration and an explanation.)

**Prove a Correspondence** - Provide a formal correspondence, using a formal reasoning system (e.g., typed lambda calculus) between the levels of abstraction.

Note: This involves proving that required properties continue to hold under the interpretation given in the formal correspondence.

**Reference Monitor** - Access mediation concept that refers to an abstract machine that mediates all accesses to objects by subjects.

**Reference Validation Mechanism** - Portion of a trusted computing base, the normal function of which is to mediate access between subjects and objects, and the correct operation of which is essential to the protection of data in the system.

Note: This is the implementation of reference monitor.

**Refinements** - Requirement in a protection profile taken to a lower level of abstraction than the component on which it is based.

Note: The refinement of a component requirement is necessary when multiple environment-specific requirements must be assigned to a single component requirement.

**Requirements** - Phase of the Development Process wherein the top level definition of the functionality of the system is produced.

**Residual Risk** - Portion of risk that remains after security measures have been applied. [NSTISSI 4009]

**Resource** - Anything used or consumed while performing a function.

Note: The categories of resources include: time, information, objects (information containers), or processors (the ability to use information) Specific examples include: CPU time; terminal connect time; amount of directly-addressable memory; disk space; and number of I/O requests per minute.

**Risk** - The expected loss due to, or impact of, anticipated threats in light of system vulnerabilities and strength or determination of relevant threat agents.

**Role** - A distinct set of operations (actions) performed on encapsulated data objects.

**Scope of a Requirement** - Determination of whether a requirement applies to: all users, subjects and objects of the TCB; all the TCB commands and application programming interfaces, to all TCB elements; all configurations, or only a defined subset of configurations.

**Security** - The combination of confidentiality, integrity and availability. [ITSEC]

**Security kernel** - An encapsulation of key security-relevant portions of an operating system that prevent unauthorized subject access to objects.

**Security Audit Trail** - Set of records that collectively provide documentary evidence of processing used to aid in tracing from original transactions forward to related records and reports, and/or backwards from records and reports to their component source transactions. [TCSEC]

**Security Relevant Event** - Any event that attempts to change the security state of the system (e.g., change access controls, change the security level of a user, change a user password). Also, any event that attempts to violate the security policy of the system (e.g., too many logon attempts). [TCSEC]

**Security Target** - Product-specific description, elaborating the more general requirements in a protection profile and including all evidence generated by the producers, of how a specific IT product meets the security requirements of a given protection profile.

**Shall** - Indication that a requirement must be met unless a justification of why it cannot be met is given and accepted.

**Should** - Indication of an objective requirement that requires less justification for non-conformance and should be more readily approved.

Note: Should is often used when a specific requirement is not feasible in some situations or with common current technology.

**Simple Security Property** - An invariant state property allowing a subject read access to an object only if the security level of the subject dominates the security level of the object.

**Specification** - one or more detailed, precise statement(s) expressing the essential characteristics of one or more facts.

**Star (*) Property** - An invariant state property allowing a subject write access to an object only if the security level of the object dominates the security level of the subject.

**State** - Give required information with no attempt or implied requirement, to justify the information presented.

**Strength of a Requirement** - Definition of the conditions under which a functional component withstands a defined attack or tolerates failures.

**Subject** - Active entity in an IT product or AIS, generally in the form of a process or device, that causes information to flow among objects or changes the system state.

**System** - IT products assembled together; either directly or with additional computer hardware, software, and/or firmware; configured to perform a particular function within a particular operational environment.

**System Entry** - Mechanism by which an identified and authenticated user is provided access into the system.

**TCB Subset** - Set of software, firmware, and hardware (where any of these three could be absent) that mediates the access of a set ***S*** of subjects to a set ***O*** of objects on the basis of a stated access mediation policy ***P*** and satisfies the properties:

(1) ***M*** mediates every access to objects in ***O*** by subjects in ***S***;

(2) ***M*** is tamper resistant; and

(3) ***M*** is small enough to be subject to analysis and tests, the completeness of which can be assured.

**Technical Policy** - Set of rules regulating access of subjects to objects enforced by a TCB subset. [NCSC-TG-021]

**Technical Security Policy** - Specific protection conditions and /or protection philosophy that express the boundaries and responsibilities of the IT product in supporting the information protection policy control objectives and countering expected threats.

**Threat** - Sequence of circumstances and events that allows a (human or other) agent to cause an information-related misfortune by exploiting a vulnerability in an IT product.

**Trace a Correspondence** - Explain a correspondence, using natural language prose, between levels of abstraction.

**Transaction** - Set of subject actions and their associated data storage accesses.

**Trap Door** - Hidden software or hardware mechanism that can be triggered to permit protection mechanisms in an automated information system to be circumvented. [NSTISSI 4009]

Note: A trap door is usually activated in some innocent-appearing manner (e.g., a special random key sequence at a terminal). Software developers often write trap doors in their code that enable them to reenter the system to perform certain functions.

**Trojan Horse** - Computer program containing an apparent or actual useful function that contains additional (hidden) functions that allow unauthorized collection, falsification or destruction of data. [NSTISSI 4009]

**Trusted Computing Base (TCB)** - Totality of protection mechanisms within an IT product, including hardware, firmware, software and data, the combination of which is responsible for enforcing a technical security policy.

Note: The ability of an organization to achieve an organizational security policy depends jointly on the correctness of the mechanisms within the TCB, the protection of those mechanisms to ensure their correctness, and on adherence to associated usage security policies by authorized users.

**Trusted Path** - Mechanism by which a person using a terminal can communicate directly with the TCB. [NSTISSI 4009]

Note: Trusted path can only be activated by the person or the TCB and cannot be imitated by untrusted software.

**Usage Security Policy** - Assumptions regarding the expected environment and intended method of IT product use.

**User** - Person or process accessing an IT product by direct connections (e.g., via terminals) or indirect connections; an individual who is accountable for some identifiable set of activities in a computer system.

Note: *Indirect connection* relates to persons who prepare input data or receive output that is not reviewed for content or classification by a responsible individual.

**User Identifier (User ID)**- Unique symbol or character string that is used by an IT product to uniquely identify a specific user.

**Virus** - Self replicating, malicious program segment that attaches itself to an application or other executable system component and leaves no external signs of its presence. [NSTISSI 4009]

**Vulnerability** - Weakness in an information system or components (e.g., system security procedures, hardware design, internal controls) that could be exploited to produce an information-related misfortune.

# ACRONYMS

| | |
|---|---|
| AIS | Automated Information System |
| CISR | Commercial International Security Requirements |
| CTCPEC | Canadian Trusted Computer Product Evaluation Criteria |
| DAA | Designated Approving Authority |
| DARPA | Defense Advance Research Projects Agency |
| DBMS | Database Management System |
| DIS | Descriptive Interface Specification |
| DoD | Department of Defense |
| FCWG | Federal Criteria Working Group |
| FIPS | Federal Information Processing Standard |
| FIS | Formal Interface Specification |
| ISO | International Standards Organization |
| IT | Information Technology |
| ITSEC | Information Technology Security Evaluation Criteria |
| LAN | Local Area Network |
| MSR | Minimum Security Requirements |
| NCSC | National Computer Security Center |
| NIST | National Institute of Standards and Technology |
| NSA | National Security Agency |
| RBOC | Regional Bell Operating Company |
| TCB | Trusted Computing Base |
| TCSEC | Trusted Computer System Evaluation Criteria |
| TDI | Trusted Database Management System Interpretation of the Trusted Computer System Evaluation Criteria |
| TRS | Travel Related Services |

# Appendix A.

# THREATS TO INFORMATION

Table 4 provides a description of common threat agents that may impact on a potential IT product (Courtney, 1991).[1] Tables 5, 6, and 7 provide common threat actions for confidentiality, integrity and availability that may occur within the environment of an IT product (Neumann, 1989). A successful threat scenario may include several threat actions in succession. The collective information in these tables also includes vulnerabilities, attacks, methods, and risks.

**Table 4. Common Threat Agents**

| | |
|---|---|
| Well-Meaning People in the Organization | Custodians, guards, users, system operators, security administrators |
| Other People in the Organization | Greedy employees, disgruntled employees, inside terrorists, intelligence agents |
| Inanimate Agents | Routine water damage (e.g., from leaking pipes), power surges and failures (e.g., from electrical storms), physical calamities (e.g., from fires, floods, civil unrest), hardware failure within the IT product, malfunctioning external devices and systems, disabled external devices and systems |
| People Outside the Organization | Hackers, hostile intelligence agents, terrorists, ex-employees |

1. According to Courtney's survey of government and commercial organizations, well-meaning employees account for 65% of reported losses; ill-intended employees account for 19%; inanimate agents (and civil unrest) account for 13%, 5% of which is routine water damage; ill-intended outsiders account for less than 3%. The nature and extent of actual losses remains unknown because incidents are not always reported.

## Table 5. Inappropriate Disclosure Threats (Confidentiality Violations)

| | |
|---|---|
| Passive Observation | Exposure (e.g., via system malfunctions)<br>Scavenging (e.g., dumpster diving)<br>Eavesdropping (e.g., on video displays)<br>Wiretapping<br>Traffic analysis<br>Analysis of IT Product emanations<br>Other forms of signals intelligence |
| Hardware Attacks | Theft of physical media<br>Physical trespass and observation<br>Implanting eavesdropping devices<br>Disarming controls (e.g., via routine maintenance) |
| Masquerade | Individuals that impersonate (e.g., via password guessing)<br>Processes that impersonate (e.g., Trojan horses) |
| Misuse of Authority | Deliberate disclosure<br>Misuse of administrative privilege<br>• Modification of access control attributes<br>• Editing of password files<br>Exploiting inference and aggregation vulnerabilities (e.g., reverse engineering)<br>Exploiting product vulnerabilities<br>• Exploiting covert channels<br>• Inadequate authentication<br>• Trap doors that bypass system checks<br>• Improper initialization or recovery<br>• Faulty reuse of objects or devices<br>• Inadequate argument validation<br>• Miscellaneous logic errors<br>• Hardware flaws<br>Browsing, searching for exploitable patterns<br>Willful neglect and other errors of omission<br>• Failing to log out when leaving a workstation<br>Preparation for misuse<br>• Code-breaking efforts<br>• Off-line password guessing<br>• Autodialer scanning<br>• Creating, planting, and arming malicious software |

## Table 6. Fault-and-Error Threats (Integrity Violations)

| | |
|---|---|
| Hardware Attacks | Implanting malicious hardware<br>Disarming hardware controls<br>Malfunctioning hardware (via aging, routine maintenance) |
| Masquerade | Individuals that impersonate (e.g., via password guessing)<br>Processes that impersonate (e.g., Trojan horses) |
| Deception of users and operators | |
| Misuse of Authority | Deliberate falsification via data entry or modification<br>Repudiation (falsely denying origin or receipt of information)<br>Misuse of administrative privilege<br>• Modification of access control attributes<br>• Editing of password files<br>Exploiting inference and aggregation vulnerabilities<br>(e.g., reverse engineering)<br>Deliberate compounding of small errors<br>Exploiting product vulnerabilities<br>• Exploiting covert channels<br>• Inadequate authentication<br>• Trap doors that bypass system checks<br>• Improper initialization or recovery<br>• Faulty reuse of objects or devices<br>• Inadequate argument validation<br>• Miscellaneous logic errors<br>• Hardware flaws<br>Willful neglect and other errors of omission<br>• Failing to log out when leaving a workstation<br>Preparation for misuse<br>• Code-breaking efforts<br>• Off-line password guessing<br>• Autodialer scanning<br>• Creating, planting, and arming malicious software |
| Lack of adequate competence | Accidental falsification via data entry or modification<br>Installing flawed application software<br>Misapplication of software<br>• Application to wrong data<br>• Miscommunication of inputs<br>• Improper runtime environment |

## Table 7. Loss-of-Service Threats (Availability Violations)

| | |
|---|---|
| Inherent system inadequacies | Inadequate deadlock avoidance<br>Inadequate response to transient errors |
| Hardware threats | Deliberate hardware modification<br>• Disabling critical components<br>• Shutting off system or power supply<br>• Implanting self-destruct devices<br>Inadvertent hardware modification<br>• Normal aging<br>• Routine maintenance<br>• Accidental damage (e.g., water damage)<br>Interference (e.g., electronic jamming, cosmic rays) |
| Usage threats | Deliberate denial of service<br>Misuse of administrative privilege<br>• Modification of access control attributes<br>• Editing of password files<br>Exploiting product vulnerabilities<br>• Exploiting covert channels<br>• Inadequate authentication<br>• Trap doors that bypass system checks<br>• Improper initialization or recovery<br>• Faulty reuse of objects or devices<br>• Inadequate argument validation<br>• Miscellaneous logic errors<br>• Hardware flaws<br>Excessive usage via masquerading<br>• Individuals that impersonate (e.g., via password guessing)<br>• Processes that impersonate (e.g., Trojan horses)<br>Creating, planting, and arming malicious software<br>Willful neglect and other errors of omission<br>Failure to order necessary supplies<br>Failure to perform routine maintenance<br>Administrative actions<br>System shutdown<br>Disabling user accounts<br>Incorrect setting of security attributes<br>Accidental deletion of critical data<br>Overload<br>Normal excess usage<br>Runaway programs<br>Personal use of organization computers |

**Bibliographic References**

[Courtney, 1991] Courtney R.H., *A Proposed Information Security Program for the NIST for Consideration by the Computer Systems Security and Privacy Advisory Board*, June 12, 1991.

[Neumann, 1989] Neumann P.G. and Parker D.B., *A Survey of Computer Abuse Techniques*, Proceedings of the 12th National Computer Security Conference, pages 396-407, October 1989.

# Appendix B.

# THE REFERENCE MONITOR CONCEPT

The concept of the *reference monitor*, "which enforces the authorized access relationships between subjects and objects of a system," was introduced by the Computer Security Technology Planning Study, conducted by James P. Anderson & Co., in October of 1972. The reference monitor concept was found to be an essential element of any product that must demonstrably implement an access control policy. The Anderson report listed three design requirements of the *reference validation mechanism*, which is "an implementation of the reference monitor concept … that validates each reference to data or programs by any user (program) against a list of authorized types of reference for that user." These requirements are:

a. The reference validation mechanism must be *tamperproof.*

b. The reference validation mechanism must *always be invoked.*

c. The reference validation mechanism must be *small enough to be subject to analysis and tests, the completeness of which can be assured.*

Early examples of the reference validation mechanism were known as *security kernels*. Security kernels typically support the three reference monitor requirements listed above. However, most commercially available systems do not implement reference validation mechanisms (e.g., security kernels) largely because their design and implementation do not fully satisfy requirement (c). General-purpose systems do not support security kernels, and their TCB generally includes key elements of the operating system and may include all of the operating system. In embedded systems, the security policy may deal with objects in a way that is meaningful at the application level rather than at the operating system level. Thus, the protection policy may be enforced in the application software rather than in the underlying operating system. The TCB will necessarily include all those portions of the operating system and application software essential to the support of the policy.

Note that, as the amount of code in the TCB increases, it becomes harder to be confident that the TCB enforces the reference monitor requirements under all circumstances. This suggests that, to demonstrably satisfy requirement (c) of the reference validation mechanism, the selection of functions to be designed within the product must be governed by the ability to completely analyze and test the reference validation mechanism. If use of state-of-the-art formal methods are required for complete analysis and test of a product, the product functions that become part of the reference validation mechanism will, by necessity, be limited in scope. For example, functions that support a wide selection of devices and access methods may not be supported. Also, access-control functions whose design and/or implementation by the reference validation mechanism are not, or cannot be, completely analyzed may limit the degree of assurance that can be obtained. Thus, requirement (c) establishes a dependency of the access control functions on the design, specification, and verification disciplines used in analysis and testing.

The concept of the reference monitor, and its implementation via the reference validation mechanism, plays the key role in supporting a wide variety of access control policies. However, the role of the reference monitor concept in other security policy areas is, by definition, limited. For example, the reference validation mechanism is not intended to implement identification and authentication policies (e.g., policies governing the choice of password complexity, strength of the encryption functions). Nor is the reference validation mechanism intended to implement availability policy (i.e., resource allocation, and fault-tolerance). Furthermore, the reference validation mechanism plays an important, but incomplete role, in establishing the penetration resistance of a TCB. Although the reference validation mechanism itself must be penetration resistant by virtue of requirements (a) and (b), penetrations caused by weak authentication or availability functions, and penetrations of privileged processes of the TCB that are not part of the reference validation mechanism, cannot be prevented by a (penetration-resistant) reference validation mechanism.

# Appendix C.

# DEFINING ACCESS CONTROL POLICIES

### Defining a Product Policy

Access control policies can be characterized in terms of five functional subcomponents, namely (1) definition of subject and object policy attributes, (2) administration of the policy attributes, (3) authorization of subject access to objects, (4) subject and object creation and destruction, and (5) object encapsulation. These subcomponents, defined in the following paragraphs, may be used to characterize a wide variety of security policies, including traditional discretionary and non-discretionary policies. The intent of characterizing all security policies in terms of these five subcomponents is to provide a general set of requirements applicable to all policies regardless of the aim of those policies and regardless of the kinds of objects controlled by those policies. These requirements provide the developers of protection profiles with a template for an access control policy component to be used in the definition of individual policies, without imposing any specific constraints on policy or on the kinds of objects involved.

Since individual policies will follow this template, combinations of policies will also be defined in terms of the five subcomponents. Whenever multiple policies are supported, these subcomponents define the composition of policies and how the policies are enforced (e.g., subject and object type coverage, precedence of enforcement). To reflect the need to satisfy all these subcomponents in each specified product policy, a single rating is assigned to access control, not to individual subcomponents. This rating is intended to capture general goals of policy specifications, instead of the differences between individual subcomponents in two or more policy specifications.

Within a policy specification, requirement can be stated as different sets of rules. These rules define the properties of each policy. Access control policy subcomponents may include requirements that may not be applicable to some policies. In such cases, the individual requirement shall be designated as non-applicable in the definition of the policy. For example, the transitive distribution of permissions applies primarily to discretionary policies. Consequently, attribute administration rules of non-discretionary controls may not include conditions for transitive distribution and revocation, and these conditions will be designated as non-applicable to a specific non-discretionary policy. Similarly, discretionary policies may not necessarily control access to object status variables (e.g., existence, size, creation, access and modification time, locking state). Hence, the rules or conditions specifying such controls may be designated as non-applicable in specific discretionary policies.

Some subcomponents may also include requirements that may not be applicable to some types of objects. In such cases, the individual requirement that is applicable to that type of object will be specified separately. The intent of providing per-type access policy specifications is to capture the access control needs of a particular type of object without imposing impractical or meaningless policy constraints. For example, user-oriented rules for access-right administration need not be imposed on objects that cannot, and are not intended to, store user data. Requiring transitive, temporal, time- and location-dependent distribution and revocation conditions for a discretionary policy on interprocess communication objects such as semaphores and sockets or on publicly accessible objects such as bulletin boards would be both impractical and unnecessary. However, when per-type specifications are used, the totality of the per-type rules and conditions must be shown to support the policy properties.

**Definition of Policy Attributes.** A policy specification must define the subject and object attributes required by that policy, and must identify the context-resident policy attributes. Subject attributes may include user-related subject credentials (e.g., user identifier, group or role identifier(s), confidentiality or integrity levels, access time intervals, access location identifier), as well as user-independent credentials assigned to privileged subjects (e.g., system privileges allowing the invocation of TCB functions unavailable to unprivileged subjects). Object attributes may include user-relevant, policy attributes (e.g., distinct object permissions for different users), as well as user-dependent attributes (e.g., secrecy or integrity levels, access time constraints, access location constraints). Finally, context-resident policy attributes may include the current time, group definitions, and/or a level indicating whether an emergency is in progress.

**Administration of Policy Attributes.** A policy specification must include rules for maintaining the subject and object attributes. The attribute maintenance rules determine the conditions under which a subject can change its own attributes as well as those of other subjects and objects. These conditions define whether a subject is authorized to modify a policy attribute and may not rely on those used in the authorization of subject references to objects (discussed below). Otherwise, a cyclic dependency may arise between the requirements of policy attribute administration and those of authorization of subject references to objects (see Chapter 7). The attribute maintenance rules also define the attributes for subject or object import or export operations.

As an example of attribute maintenance rules, consider those rules that determine what subjects have the authority to distribute, revoke, and review policy attributes for specific subjects and objects, and the conditions under which these actions can be performed. The distribution and revocation rules determine which of the following conditions are enforced.

a. **Selectivity:** distribution and revocation can be performed at the individual attribute level, such as user, group, role, permission, privilege, security or integrity level.

b. **Transitivity:** a recipient of a permission from an original distributor can further distribute that permission to another subject, but when the original distributor revokes that permission from the original recipient, then the subject which received that permission from the original recipient will also have it revoked.

c. **Immediacy:** the effect of the distribution and revocation of policy attributes should take place within a specified period of time.

d. **Independence:** two or more subjects can distribute or revoke policy attributes to the same subject independent of each other.

e. **Time-dependency:** the effect of the distribution and revocation of policy attributes must take place at a certain time and must last for a specified period of time.

f. **Location-dependency:** the distribution and revocation of policy attributes must take place at a certain location.

The review rules determine which of the following two kinds of review are supported and impose conditions constraining the review of attributes.

1. **Per-object review:** for an object, list all (or a specified class of) attributes that govern the relationship between that object and a specified set of subjects that may directly or indirectly access that object.

2. **Per-subject review:** for a subject, list all (or a specified class of) policy attributes which govern the relationship between that subject and a specified set of objects that subject may directly or indirectly access.

The imposed conditions for allowing the review of attributes determine, in particular, which users of an object may discover which users have access to that object, as well as what subjects may be used to access that object.

The coverage of attribute-review rules is specified in terms of the kinds of objects and subjects to which they apply. If different rules and conditions apply to different subjects and objects, the totality of these rules must be shown to support the defined control objectives. If a composition of several policies is to be supported, attribute administration must be composed.

**Authorization of Subject References to Objects.** A subject's reference to an object consists of invoking an action on a set of objects. The subject's reference to the object can be thought of as a request to access that object. Examples of actions include invocations of TCB commands, function calls, processor instructions, protected subsystems, and transactions. An action may have separate policy attributes from those of the issuer of the reference. For example, invocations of transactions and protected subsystems (which encapsulate objects) will generally include policy attributes that differ from those of their invokers. In contrast, other actions such as invocations of individual processor instructions, TCB function calls, some TCB commands, and applications programs are prohibited from using policy attributes, such as identity, group, role, or secrecy and integrity levels, that differ from those of their invoker. Policy attributes involved in rules for deciding access authorization are referred to as "access control" attributes.

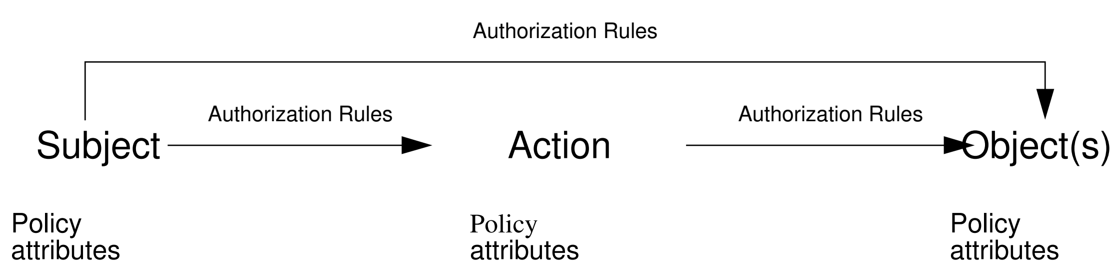



**Figure 14. Authorization of Subject References to Objects**

The rules for authorizing subject references to objects are defined in terms of (1) the subject's authorization to an action, (2) the action authorization to one or more objects, and (3) the subject's authorization to one or more objects, as illustrated in Figure 1. These rules are based on the policy attributes defined for subjects and objects. The rules are defined either on <subject, action> and <action, object(s)> tuples or on <subject, action, object(s)> triples, depending upon the specified policy. The authorization rules specify the authorization scope and granularity in terms of (1) resources containing one or more objects, (2) individual subjects and objects, (3) the subject and object policy attributes, and (3) the subject and object status attributes (e.g., existence, size, creation, access and modification time, locked/unlocked). The authorization rules also specify whether delegated authorization (i.e., authorization of a subject access performed on behalf of other subjects, using combined-subject attributes) is allowed.

The coverage of the authorization rules is specified in terms of the types of objects and subjects to which they apply. If different rules apply to different subjects and objects, the totality of these rules is shown to support the defined policy properties. If multiple policies are supported, these rules define the composition of policies and how the authorization conditions are enforced (e.g., subject and object type coverage, order of enforcement)

**Creation and Destruction of Subjects and Objects.** The rules for allowing the creation and destruction of subjects and objects must be defined. These rules impose the following conditions under which subjects and objects can be created and destroyed.

a. **Creation and destruction authorization**: the authorization of specific subjects to create and destroy a subject or an object and with what attributes.

b. **Object reuse**: the revocation of all authorizations to the information contained within a storage object prior to initial assignment, allocation or reallocation of that storage object to a subject from the TCB's pool of unused storage objects; no information, including encrypted representations of information, produced by a prior subject's actions should be available to any unauthorized subject.

c. **Space availability**: the capacity and presence of storage space shall be available for the creation of a subject or object.

d. **Definition of default attributes subject** or the default values and rules for inheriting object attributes, if any, shall be defined.

**Object Encapsulation.** The encapsulation subcomponent of an access control policy specifies that a subject's access to a objects be constrained in such a way that (1) all accesses to these objects occur via access to a logically and/or physically isolated set of subjects that protect these objects from more general forms of access, with each subject having a unique protected entry point; and (2) confinement of this set of protecting subjects is such that these subjects cannot access any other objects and cannot give away access to the objects they protect.

Discretionary encapsulation allows individual (privileged and unprivileged) users to create protected subsystems and to set access to them at their own discretion (perhaps using well-known discretionary access control mechanisms). Non-discretionary encapsulation uses logical and/or physical domains (and perhaps security levels) to enforce encapsulation at the product level (i.e., by system administrators as opposed to at the discretion of the creator of the protected subsystem). The traditional DoD mandatory policies may be useful for encapsulation in some environments. For example, one could use DoD mandatory policies to encapsulate a protected subsystem by reserving a sublattice of compartments for the programs and data objects of that subsystem. (Some trusted database management systems use this approach for the support of per-client Database Management System (DBMS) servers. The server(s) and database objects are encapsulated in a reserved sublattice of the TCB). Note that both discretionary and non-discretionary encapsulation can involve the use of surrogate subjects to protect the entry points to protected subsystems.

The rules for object encapsulation must be defined whenever object encapsulation is supported. The rules for object encapsulation constrain (1) access authorization to encapsulated objects (i.e., a subject access to an object can take place only if the subject invokes another subject that performs the requested action on the object using additional authorizations associated with the encapsulation); (2) application-level encapsulation (i.e., they define conditions for the creation of encapsulated subsystems); and (3) invocation of encapsulated subsystems.

### Composition of Access Control Policies within a Product

Many of the access control policies supported by a product represent a composition of two or more basic access control policies. The need to compose basic policies arises for at least two reasons. First, to extend the range of an IT product's protection applicability, new applications subsystems or individual functions may be added to a TCB. These subsystems and functions may support different basic access control policies from those supported by the original TCB. These different policies must be composed with those of the original TCB. Second, to support new system or organizational policies, functions implementing new basic access policies are required to be added to a product's TCB. These new access control policies must also be composed with the existing ones to enable the implementation of the protection objectives of an organization.

The composition of access control policies within a product adds new requirements to the definition of product access control policies. For example, whenever trusted subsystems or functions that extend the TCB are added to support new policies, it must be ensured that existing TCB functions can not be used to access the new subjects and objects in an unauthorized way, and that the new subsystems and functions can not be used to access the currently existing subjects and objects in an unauthorized way. Also, whenever multiple policies are composed within the same TCB and refer to the same set of subjects and objects, it must be determined that the composition of access control policies is consistent with the overall TCB protection policy and does not introduce new vulnerabilities.

The composition of access control policies within an access control component also requires that both the individual access control policies and their rules for composition be completely defined (i.e., for each element of the defined policy, a corresponding set of rules must establish the completeness of the composition).

**Composition of Discretionary and Non-Discretionary Policies.**

A typical example of access control policy composition within the same IT product TCB is provided by the addition of a non-discretionary access control policy (e.g., the DoD mandatory policy) to a TCB that originally supports only a discretionary policy. The composition rules for the resulting TCB access control policy require that (1) both the mandatory and discretionary authorization rules be enforced on every subject and object protected by discretionary controls, and (2) the references issued by the enforcement modules of the discretionary policy be subject to the mediation specified by the mandatory rules. This precedence of enforcement is important whenever the exceptions returned by the enforcement of the two sets of rules are different. The reason is that if non-identical exceptions are returned by the two sets of rules, new covert channels may appear that would not appear had only the mandatory rules be enforced. These covert channels would violate the intent of the mandatory secrecy policy.

Other examples of policy composition within the same TCB include those in which the DoD mandatory secrecy policy and a mandatory integrity policy are supported. This composition might imply (1) that both the mandatory authorization rules be enforced on every subject and object reference and (2) that the controlled sharing rules of the two mandatory policies must be compatible with each other. Compatibility of these rules would imply, for example, that the secrecy and integrity upgrade conditions must not introduce covert channels that otherwise would not exist when the individual policies were used separately.

**Composition by Policy Partitioning.**

A typical example of policy partitioning appears when a subsystem implementing its own access control policy is integrated within an operating system TCB. (An alternate way of integrating such a subsystem in a trusted operating system is illustrated in the following discussion of TCB policy subsetting). Such subsystem integration is fairly common of database management systems and products. Since these subsystems implement their own policies, which generally differ from those of the operating system, the composition must ensure that neither the operating system nor the database subsystem interfaces of the same TCB would allow (1) an untrusted database application or an unprivileged database user to access operating system objects in an unauthorized manner, or (2) an untrusted operating system application or an unprivileged operating system user to access database objects in an unauthorized manner. Furthermore, when non-discretionary access controls are implemented in both the operating system and the database subsystem, the composition of the two should not introduce covert channels that were not present when the individual policies were supported.

The suggested composition causes the access control partitioning of the TCB into an operating system and a database partition. The two partitions can share other TCB policy components such as identification and authentication, system entry, and trusted path. Other similar examples of policy partitioning are offered by message or mail subsystems and communication protocol subsystems.

**Composition by Policy Subsetting.**

An alternate method of policy composition is that provided by policy subsetting. In this method, separate TCB subsets are allocated different policies. This method of policy composition is addressed in detail in the *Trusted Database Management System Interpretation of the Trusted Computer System Evaluation Criteria* (TDI) [NCSC 1991].

In this composition method a TCB subset, M, is a set of software, firmware, and hardware (where any of these three could be absent) that mediates the access of a set of subjects, S, to a set of objects, O, on the basis of a stated access control policy, P, and satisfies the properties or the reference validation mechanism [NCSC 1991]. M uses resources provided by an explicit set of more primitive TCB subsets to create the objects of O, create and manage its data structures, and enforce the policy P. (The above definition does not explicitly prohibit an access control policy P that allows trusted subjects.) If there are no TCB subsets more primitive than M, then M uses only hardware resources to instantiate its objects, to create and manage its own data structures, and to enforce its policy. However, if M is not the most primitive TCB subset, then M does not necessarily use the hardware or firmware functions to protect itself. Rather, it uses either hardware resources or the resources provided by other, more primitive TCB subsets. Thus TCB subsets build on abstract machines, either physical hardware machines or other TCB subsets. Just like reference validation mechanisms, a TCB subset must enforce a defined access control policy separately than those enforced by other subsets.

The access control policy P[i] is the policy allocation for each identified TCB subset M[i] of a product along with the relation of these policies to the product policy P. The allocated policies P[i] will be expressed in terms of subjects in S[i] and objects in O[i]. To satisfy the requirement that the (composite) TCB enforce its stated policy P, each rule in P must be traceable through the structure of the candidate TCB subsets to the TCB subset(s) where that enforcement occurs. It must also be noted that every subject trusted with respect to P[i] must be within the TCB subset M[i].

**An Example of Organizational Protection Objective: Separation of Roles**

Separation of roles provides for the compartmentalization of authority and responsibility, and reduces the potential damage from errors or accidents or damage caused by unskilled or corrupt users or administrators. Separation of roles facilitates the secure administration of a product by enabling administrators to distribute, review, and revoke permissions of users to objects based on individual roles instead of on individual users and individual objects. This objective appears to be very common in organizations that have a stable structure with well-defined roles and that frequently change which users fill which roles.

The separation-of-role policies provide a measure of data confidentiality and integrity by reducing the likelihood of an unauthorized action being taken, or limiting the effects of an unauthorized action if it does occur. To accomplish this, these policies must associate (1) identified and authenticated users with roles; (2) roles with sets of identified actions (e.g., executions of specific TCB functions, commands, application programs, and transactions identities differing from those of their invokers); and (3) identified actions with sets of objects. The first two associations must be non-discretionary whereas the third can be either discretionary or non-discretionary, depending upon the threats assumed in the product's environment (e.g., whether Trojan horses or viruses are assumed to be included in the code of a transaction).

An Access Control Policy component providing fine-grained separation can be used to restrict the capabilities of an unskilled or corrupt administrator to more specific duties. By creating audit administrator, account administrator, and operator roles (for example), potential damage can be contained and the integrity of the product and its resources and data can provide greater protection. The functions performed by the various security-relevant roles (e.g., security administrator) are identified. The administrative personnel should be able to perform security administrative functions only after taking a distinct auditable action to assume a security administrative role on the product. Non-security functions that can be performed in a security administration role should be limited strictly to those essential to performing the security role effectively.

Complete separation of roles is provided in those products which, in addition to separate administrator roles, provide the ability to define roles for subjects. For example, a bank would require distinct role definitions for bank tellers, loan officers and clerks. Different definitions would also be required on a military supply system for order clerks, shipping and receiving personnel, and combat unit personnel.

# Appendix D.

# MODULAR DECOMPOSITION

This Appendix discusses the notion of TCB modular decomposition and how the specific assurance requirements for modular decomposition can be satisfied. First, a definition of a module is given. The discussion on decomposition relations gives background information on how to satisfy the requirements of level MD-2 for the identification of the protection functions and the interface between the modules through the use of the *contains* and the *uses* relations. Finally, a discussion on correctness dependencies among modules gives background information on how to satisfy the requirements of level MD-3 for an analysis of the correctness dependencies (i.e., for service and environment dependencies).

**Module Definition.** A module is defined as an independently replaceable product part (unit, building block). A software product module has the following five characteristics: a *role*, a *set of related functions*, a *well-defined interface*, an *internal design*, and *replacement independence*.

1. **Role.** A module has a well-defined unique role (responsibility, purpose, contract) that describes its effect as a relation among inputs, outputs, and the retained state of the module. The role of a module describes its effects or behavior on inputs. The effects can be reflected in the values of outputs or the retained state of the module. The state of a software module can be represented by a set of variables (simple variables or tables). A well-defined role should have a short and clear description. A module should have a simple name that reflects its role. Typically, module roles are product unique; no two modules of a product have the same role (no duplication of roles). However, the product may intentionally duplicate modules to achieve other product goals (e.g., performance, reliability).

2. **Set of Related Functions.** A module contains only all the functions (procedures, subroutines) necessary to satisfy its role. Each function has well-defined inputs, outputs, and effects. Functions can, but need not, be named. In software, for example, some functions are expanded in-line for performance reasons; however, such in-line expansion may not be expressible in some programming languages. The name of a function should

reflect its purpose. The inputs and outputs of a software function can be formal parameters, informal (global, environment) parameters, or (request-response) messages. It should be simple to distinguish the public from the private functions (if any) in a module. It is desirable, but not necessary, that the functions of a module be nonredundant (function redundancy is at the discretion of the product or module designer). Regarding the “all and only” nature of a module’s functions, certain functions typically have a complementary twin, (e.g., get-set, read-write, lock-unlock, do-undo, reserve-unreserve, allocate-deallocate).

3. **Well-Defined Interface**. A module interface is well-defined if it contains all and only the module assumptions that a module user needs to know. A module has an interface (external specification) that consists of the public (visible) items that the module exports. For software, a well-defined interface contains declarations of exported (public) functions and their formal parameters, data, types, manifest constants, exceptions raised, exceptions handled, exception handlers, and, the associated rules (restrictions or discipline) for using these public functions, types, constants, and global variables. The discipline of an interface, if any, may explain or constrain the “legal” order in which to use public functions. However, it may be inappropriate or impossible to capture certain programming restrictions or disciplines. In such cases, the restrictions should be provided in associated documentation or commentary. Note that a module interface includes variables that are global to that module.

4. **Internal Design.** A module has an internal design that contains its construction assumptions and details how its interface is satisfied. It should be possible to understand the interface (and role) of a module without understanding its internal design. Module users need not know the module internal design.

5. **Replacement Independence.** A broken or non-functional module can be replaced (e.g., with a corrected function having an identical interface) without also replacing any other module in the product. In software products, the notion of replacement independence has a somewhat different meaning. While replacement independence is implied by information hiding, and information hiding disallows global variables, replacement independence does not rule out the use of global variables in software modules, provided that the global variables are explicitly defined in the module’s interface, and that the dependencies among the modules using those global variables are known.

**Decomposition Relations.** The decomposition of any product into modules relies on two intermodule relations, namely (1) the *contains* relation, and (2) the *uses* relation. These relations imply certain correctness dependencies among modules that are fundamental to the understanding of the module structure of a product.

1. **The Contains Relation**. Internally, a module may contain component submodules. Sometimes it is necessary or desirable to identify a set of component parts of a module as submodules. These submodules partition the parent module in a collectively exhaustive and mutually exclusive manner. The decision as to when to stop partitioning a product into additional modules is generally based on a designer's discretion and economics (i.e., no apparent return on the effort) as there is no other generally accepted criterion for when to stop.

Collected product-wide, the *contains* relation yields a module hierarchy (tree). Nodes of the tree represent modules. Arc (*A*, *B*) means that module *A* directly contains submodule *B*. The root is the zero-th level of the tree. The product itself should be considered as the zero-th level module. The (n+1)-th level consists of the children (direct submodules) of the n-th level. Modules with no submodules are called leaf modules. We can define a part hierarchy product as modular if the product itself, and recursively each of its subparts that we identify as should-be-modules, satisfies the definition of module.

2. **The Uses Relation.** We define the uses relation between functions and modules as follows. Function *A* uses function *B* if and only if (1) *A* references *B* (e.g., *A* invokes *B*, *A* reads data written by *B*) and uses results or side-effects of that reference and (2) there must be a correct version of *B* present for *A* to work (run, operate) correctly. A function uses a module if and only if it uses at least one function from that module. A module uses another module if and only if at least one function uses that module. The uses relation is well-defined. From the *uses* relation we can draw a directed graph for a given level, where the nodes are same-level modules, and arc (*A*, *B*) means that module *A* uses module *B*. Also, we can draw a *uses* graph of the leaf modules.

**Correctness Dependencies Among Modules.** Correctness dependencies among modules are basic to describing, evaluating, and simplifying the connectivity of modules, and therefore basic to product review and restructuring. For modules *A* and *B*, *A* depends on *B*, or *A* has a correctness dependency on *B* or "the correctness of *A* depends on the correctness of *B*", if and only if there must be a correct version of *B* present for *A* to work correctly. There are two types of correctness dependencies.

1. **Service Dependency.** A service dependency exists where *A* references a service in *B* and uses results or side-effects of that service. The service may be referenced by invocation through a function call, message, signal (e.g., a semaphore or monitor operation), or hardware trap, or by sharing data. It is important to point out that not all invocations are service dependencies. For example, some services invoke other services simply to provide notification or advice of the occurrence of certain events and do not rely on the results of the invoked service. In contrast, data sharing always represents service dependencies. Modules that are either *readers* or *writers* of shared data depend on other modules that are writers of those same shared data. Thus, shared data with multiple writer modules produce mutual dependencies and increase module connectivity.

2. **Environmental Dependency.** An environmental dependency may exist even though *A* may neither invoke nor share any data with *B*, but nevertheless depends upon *B*'s correct functioning. Examples of environmental dependencies are provided by the dependencies of most of a product's services on the interrupt, signal, and global exception handling subsystems. Other low-level services may become part of the "environment" of all higher-level services (e.g., recovery services) and, thus, environmental dependencies become pervasive in all products.

For structural analysis, it is desirable to represent correctness dependencies between product modules with the *contains* and the *uses* relations and their graphs. As seen previously, the *contains* relation among modules is unambiguously defined by syntactic analysis. In contrast, the uses relations can be defined in two possible ways: (1) as representing all correctness dependencies; or (2) as representing only service dependencies. To use all correctness dependencies is desirable but impractical in all but small products. Hence, the uses relation could be defined in terms of only service dependencies. To simplify product structure, we need to minimize correctness dependencies and eliminate all circular dependencies. To do this, we first minimize data sharing dependencies, because they contribute to circular dependencies, then we remove other circular dependencies. The achievement of the product simplification goal can be measured by the results of eliminating global variables and acyclic structure, and minimizing the cardinality of the *uses* relation. If this *uses* relation represents all product correctness dependencies and if its graph is cycle-free, then showing correctness of the product parts in a bottom up order (the reverse of a topological sort of the *uses* graph) leads to correctness of the product.

# Appendix E.

# PENETRATION ANALYSIS

This appendix discusses the notion of penetration analysis and how the specific assurance requirements for penetration analysis are satisfied. First, the scope of penetration analysis is defined. This gives background information for the level PA-1 requirement to define the TCB configuration, interface, and protection functions that are subject to penetration testing. Next, the precision coverage and test conditions for penetration analysis are discussed. This gives background information for the level PA-2 requirement on how the system reference manuals, DIS, and source code are used to define the penetration coverage and test conditions. Penetration resistance properties are then discussed which gives background information for the levels PA-3 and PA-4 requirements regarding the derivation and verification of penetration resistance properties.

Penetration analysis requires that the *scope* of the analysis method be defined. This includes the following requirements: (1) that the product and TCB configuration be defined and frozen, (2) that the TCB protection functions that are the subject of analysis be identified, and (3) that the TCB interface that is subject to penetration be defined. The TCB configuration includes both the hardware and the software configuration. The TCB protection functions include, but are not restricted to, the identification of the security policies supported, reference mediation, and TCB protection components. The TCB interface includes all the unprivileged user-visible and application programming interfaces. The user-visible interfaces may also include privileged user interfaces, depending upon whether the vulnerabilities of the security management and administrative roles need to be analyzed.

Penetration analysis also requires that the *precision* and *coverage* of the analysis method be defined. The analysis precision depends on several factors, including the level of analysis detail, and the definition of the types of TCB interface, design, and implementation documentation used. Typically, the TCB documentation includes the programming reference manuals—not just the TCB interface definition, DIS and FIS—and source code.

The coverage of the analysis methods includes the goals, or purposes, and the outcomes of the penetration method. Among the typical goals of penetration analysis are flaw identification and repair, vulnerability assessment, and testing of known classes of penetration flaws found in other TCBs that might be relevant to the particular product's TCB. These goals are usually defined in terms of a set of threats deemed important to counter for the product's security. The outcomes of the penetration analysis are the identification of security flaws and the demonstration of the effects of those flaws. A common desirable outcome of penetration analysis is the confidence that a directed, comprehensive examination of the TCB has been performed by skilled analysts using state-of-the-art methods and tools for a given period of time. This is a generally useful outcome even when few flaws are discovered.

Penetration analysis methods include penetration testing and verification of penetration-resistance conditions. Penetration testing requires that test conditions, or flaw hypotheses, be generated, and test data (e.g.,test environment, TCB call parameters, and outcome) are defined. Test conditions are typically generated from existing TCB documentation, from known generic flaws, from known flaws of other similar products, and from implementation inspection. Note that the test conditions need not include policy-oriented conditions; those are the subject of the security functional testing. Penetration testing also requires that the tests be carried out to confirm that a particular flaw is, in fact, real. In some cases, the test need not be carried out if the design analysis confirms the existence of a flaw, or if penetration scenarios for that flaw exist or were confirmed by testing on a similar product.

In contrast with penetration testing, which rests on the hypothesis that penetration flaws exist, the penetration-resistance verification is based on the hypothesis that a TCB achieve various degrees of penetration resistance whenever it adheres to a specific set of design properties. In particular, these design properties are derived by interpreting the requirements for reference mediation and TCB protection functions. Note that, as in the case of penetration testing, the penetration resistance properties need not include security policy properties; those are the subject of the security policy verification. For example, whether the authentication function of an identification and authentication component satisfies its definition is a concern of policy verification. Whether such a function resists a dictionary attack is a concern of penetration resistance.

The set of penetration-resistance properties refer to, but are not restricted to, the following functional components: (1) TCB isolation (or tamper proofness), which includes call parameter validation, TCB/user space separation checks, control of both TCB entry-point checks and TCB privilege checks; (2) TCB noncircumventability, which guarantees that all references to TCB object (e.g., references to object status variables, object permissions) are mediated; (3) consistency of TCB global variables and objects, which maintains the properties of global variables, objects, and internal functions of the product; (4) timing consistency of condition (validation) checks, which assures that the validity of a condition (validation) check is not lost at the moment when an action that depends on that check is actually performed; and (5) elimination of undesirable system and/or user dependencies, which ensures that unnecessary dependencies between system and user are not present in the system.

Unlike flaw hypotheses, the penetration-resistance properties are captured in penetration-analysis models by the model constants and the state-transition rules. A model must be based on a penetration resistance policy. For example, the policy may state that a TCB element may be altered or viewed, or a TCB internal function may be invoked, only if the set of conditions associated with the alter/view/invoke access specified by penetration-resistance properties are validated in an atomic sequence (with the alter/view/invoke operation itself); i.e., if the timing consistency of the condition check is preserved. Penetration-resistance modeling is useful for the same reasons as those for security policy verification, namely, it enhances the understanding of the penetration resistance properties and enables the design of verification methods and tools. In the context of penetration resistance, models suggest that penetrations, which are caused by incorrect implementation of the penetration-resistance properties within a TCB, can be identified in a TCB's source code as patterns of incorrect or absent validation-check statements or flaws that violate the intended design specifications or source code.

# Appendix F.

# MOTIVATION FOR DEPENDENCY ANALYSIS

This appendix illustrates the need for dependency analysis in the development of protection profiles, and classifies functional and assurance dependencies.

The analysis of dependencies among the functional and assurance components of a profile is necessary for at least the following four reasons. Dependency analysis helps (1) avoid inadequate, or incorrect, profile specification; (2) avoid overspecification of a profile; (3) determine the effect of profile changes (e.g., addition or removal of individual components or component requirements); and (4) analyze the compatibility of different protection profiles (e.g., for the harmonization of different security standards).

1. **Avoiding inadequate, or incorrect, profile specification.** Inadequate, or incorrect, profile specification can manifest itself in many guises. One manifestation is that of *missing specification*s of functional or assurance requirements that are necessary to achieve the goal of a particular protection profile. For example, profiles might not include either requirements for TCB physical protection or requirements for operational, administrative, or environmental protection that could compensate for lack of TCB physical protection. Other profiles may include information flow (i.e., covert-channel) identification and control as part of the confidentiality policy components but omit them from the integrity policy components, or may include trusted recovery as part of the integrity components but omit them from the confidentiality components.

Another manifestation of inadequate specification is that of including *insufficient or incomplete requirements* in a profile. For example, requirements of TCB modularity and TCB module support of the least privilege principle are meaningless without a definition of the module. And, structuring the TCB into largely independent modules cannot be performed unless the identification of intermodule relationships is required. Other profile requirements may become insufficiently specified whenever they are overly abstract and general and cannot be used for specific components. For example, in a window-based product, many of the general device-labeling requirements lack specifications for input devices (e.g., mouse, keyboard) that operate across independently labeled windows.

Inadequate requirement specifications can manifest themselves as *inconsistent requirement levels,* (e.g., inconsistent specification scope, granularity, coverage, or strength for different functional components). For example, when the profile also requires the handling of covert storage channels, the scope and granularity of non-discretionary policies of a profile cannot be used at a level that requires only the control of access to a defined subset of subject and objects, or to the object contents, but not object status attributes. The handling of covert storage channels (e.g., elimination, bandwidth reduction, audit) would become immaterial when the scope and granularity of non-discretionary policy would make overt means of data leakage available to unprivileged programs or users. Similarly, the TCB structuring requirement of minimizing the TCB complexity by excluding from the TCB all modules that are not protection-critical would be inconsistent with a level of modularity that requires only subsystem-level TCB decomposition. Such a structuring requirement would also be inconsistent with a level of modularity that requires only modular decomposition but does not require the analysis of inter-module relationships. In both cases, the requirement inconsistencies makes it impossible to determine the protection-criticality of a module targeted for exclusion from the TCB.

Inadequate specifications can cause *cyclic dependencies* among the requirements of functional or assurance components of a profile. A cyclic dependency appears between two (or more) requirements whenever they mutually depend on each other. For example, a cyclic dependency may arise in the specification of the controlled sharing requirements and the access authorization requirements. To authorize access to an object, the controlled sharing components of access control would have to modify a separate object or an object access-control attribute. This modification action, in turn, requires access authorization. Cyclic dependencies may appear in an incompletely specified profile definition or may be introduced in a profile definition when individual requirements of functional components are refined, or elaborated, and/or when new functional-component requirements are added to the profile definition. Cyclic dependencies among profile requirements are particularly undesirable both because they make it difficult to reason about the profile consistency and completeness and because they make it difficult to demonstrate that such requirements are satisfied in practice. Consequently, cyclic dependencies among requirements must be identified and, whenever possible, eliminated (see the example of cyclic dependency removal in the next section).

Finally, inadequate profile specification may, in fact, impose requirements that are *demonstrably impossible* to satisfy in general. For example, a general access review specification requiring the determination of whether a subject could obtain a certain permission to an object cannot, in general, be satisfied by discretionary access control models; viz., the uniform safety problem [Denning83]. Only some specific discretionary policy implementations could allow such review features and, therefore, the requirement would have to be restated in terms of the definition of a specific policy implementation.

Dependency analysis helps avoid inadequate, or incorrect, profile specification. By requiring that the different types of dependencies among the functional and assurance components be identified (discussed below), the analysis helps determine whether any necessary requirements are missing from a profile definition. By establishing a transitive closure of all dependent requirements, the analysis helps determine whether incomplete or inconsistent levels of requirements are specified in the profile. The transitive closure of all dependent requirements implies that cyclic dependencies have been identified and removed.

2. **Avoiding overspecification of a profile.** The overspecification of protection-profile requirements may limit the practical usefulness of the profile by levying unnecessary functional and/or assurance requirements on products. It may also inadvertently *bias the profile* towards specific products or environments of use. This would be particularly inappropriate since profiles are intended to capture the essential protection requirements of diverse products and environments. For example, a profile whose functional or assurance components require that segmentation be supported by the hardware to represent storage objects biases the profile towards products that support segmented address spaces and against products that support linear address spaces (e.g., in reduced instruction set architectures). A profile that defines what privileges must be used in a TCB also biases the profile towards a specific product. Similarly, a profile may be undesirably biased towards a specific environment. For example, if a profile includes a specific set of mandatory authorization rules instead of generic access control rules, the profile may become biased to a non-commercial environment in which classified information is processed. This bias may be desirable if the environment of profile use is appropriately recognized as being restricted.

Overspecification may add *impractical requirements* to a profile, namely, requirements that could not be satisfied by most, or even any, product, in practice. For example, a requirement of enforcing the same access authorization rule on all types of objects would be impractical because, in most systems, access control rules are implemented on a per-object-type basis. The authorization rules for directories differ from those on shared memory segments and message queues, and authorization rules for database objects differ from those for operating system objects. A further example of overspecification is requiring flexible discretionary access control policies for user-oriented controlled sharing on objects that cannot, and are not intended to, store user data, (e.g., requiring access control lists on interprocess communication objects and on user processes as opposed to the files containing the programs executed by user processes). For such objects, simple controlled sharing rules that isolate the object from unwanted or unintended sharing would be adequate.

Profile overspecification may cause *meaningless requirements* to be levied on a product. Functional component requirements may be meaningless for products that cannot, and are not intended to, support certain functions and operations. For example, functions that prevent object reuse are irrelevant to products that only need to support a fixed number of objects that are never destroyed. Similarly, TCB recovery functions and self-checking functions for disk data structures are irrelevant to diskless workstations.

Dependency analysis helps avoid profile overspecification. By establishing a transitive closure of all dependencies among the functional and assurance components and between the security functional components and operational characteristics of various products, the analysis helps determine whether unnecessary, impractical, or meaningless requirements, or levels of requirements, are specified in the profile.

3. **Determining the effects of profile changes.** Registered protection profiles are expected to be stable for extended periods of time (e.g., years and possibly decades). However, as experience accumulates with building security targets and products using these profiles, the need for incremental changes and maintenance of profile components becomes inevitable. Such changes may affect both individual profiles and profile families. The fundamental concerns that arise in making such changes are those of determining the effect of a requirement change on the balance of the profile requirements and of maintaining the consistency and coherence of the changed profile. These concerns are also relevant in determining whether the updated profile should be accepted into the profile registry.

For example, suppose that the security-testing component of a profile is changed to require explicit, systematic test-coverage analysis and coverage documentation in test plans. This change implies that, regardless of the coverage analysis method, the test conditions must be generated from the interpretation of the policy models in the TCB. Otherwise, the notion of test coverage would remain undefined. Hence, this change suggests that a requirement for policy model and its interpretation must be included in the assurance (i.e., development process) components of the profile (family). Furthermore, if a more specific form of coverage is required, additional requirements may become necessary. For instance, if either data-flow or path coverage is specified, then the identification of the protection-critical modules becomes necessary because these types of coverage require that a graph of all authorization checks must be derived from the source code. Hence, both modular decomposition at a level that provides intermodule correctness dependencies and implementation requirements must be included in the development process components of the profile (family).

Dependency analysis is a primary requirement for determining the effect of profile changes and maintenance. By establishing a transitive closure of all dependent requirements, the analysis helps determine precisely the scope of the profile change in terms of the required components and levels of required components.

4. **Analyzing the compatibility of different protection profiles.** An important objective of this standard is to enable the comparative analysis of product security requirements of existing standards with those protection profiles accepted under this standard. Several national and international security standards already exist, and a substantial number of different products have been evaluated under these standards. Thus, it is important to establish relationships between the existing standards and evaluated products, and the accepted profiles of this standard.

An impediment that arises in any attempt to establish a relationship among the profiles of two different standards is the fact that some components are classified as assurance components in some standards and as functional components in others. For example, the TCSEC includes operational assurances that are classified as functional assurances in the CTCPEC and the ITSEC. Although this may only be considered a superficial problem of profile organization, in practice it may have far reaching implications because the rating of the functionality levels may not be based on the same parameters as those of the assurance levels in different standards. Thus, the ability to establish a correspondence between, and harmonize, profiles of different standards may be hampered by inconsistent requirement classification. Furthermore, establishing the correspondence between these profiles requires that the dependencies among the components of both individual profiles be analyzed and preserved under the established correspondence.

Dependency analysis is an important step in establishing the correspondence between profiles of different security standards. This analysis decreases profile susceptibility to inconsistent classification of a component either as a function or as an assurance. Regardless of how a component is classified, its dependencies with other components of the same profile are identified by dependency analysis. Those dependencies could then be preserved under the established correspondence.

# Appendix G.

# EXAMPLE ASSURANCE PACKAGES

This appendix provides seven example assurance packages using the development assurance components defined in Chapter 5 and the evaluation assurance components defined in Chapter 6. The structure of each assurance package follows that of the development assurance and evaluation assurance components, (i.e., each package contains components, where required, for development process, operational support, development environment, development evidence, and from the classes of evaluation methods: testing, review and analysis). The designations T1 through T7 indicate a ranking of assurance provided by the package. These packages are summarized in Table 15 at the end of this appendix.

**Assurance Package T1**

The T1 assurance package is intended to include IT products which meet the minimum assurance levels acceptable to consumers whether in the DoD and Intelligence Community or the commercial world. This assurance package includes only the developmental assurance and evaluation assurance components, at their lowest levels, deemed necessary to provide minimal understanding of the product.

The intent of the development process assurance for this package is to establish that the external behavior of the product conforms to its user-level and administrative documentation without any analysis of the internal structure of the IT product's TCB. For this reason, only the claimed TCB protection properties, the TCB element list, and the TCB interface description are required to enable functional testing.

The intent of the operational support assurance for this package is to establish a minimal level of user and administrative guidance and product information that enables the correct product installation and use of the product's protection functions.

There are no requirements in this package pertaining to the environment in which the product is developed.

The intent of this package is to require the type of assurance evidence that was generated during the normal development process of a product that was rated C2 by TCSEC standards. Table 8 summarizes the generic assurance components that comprise the T1 assurance package.

**Table 8. T1 Assurance Package**

| Assurance Component Summary | T1 |
|---|---|
| **Development Assurance Components** | |
| **Development Process** | |
| **TCB Property Definition** | PD-1 |
| **TCB Design** | |
| TCB Element Identification | ID-1 |
| TCB Interface Definition | IF-1 |
| TCB Modular Decomposition | ---- |
| TCB Structuring Support | ---- |
| TCB Design Disciplines | ---- |
| **TCB Implementation Support** | ---- |
| **TCB Testing & Analysis** | |
| Functional Testing | FT-1 |
| Penetration Analysis | ---- |
| Covert Channel Analysis | ---- |
| **Operational Support** | |
| **User's Security Guidance** | UG-1 |
| **Administrative Guidance** | AG-1 |
| **Trusted Generation** | ---- |
| **Development Environment** | |
| **Life Cycle Definition** | ---- |
| **Configuration Management** | ---- |
| **Trusted Distribution** | ---- |
| **Development Evidence** | |
| **TCB Protection Properties** | EPP-1 |
| **Product Development** | EPD-1 |
| **Product Testing & Analysis** | |
| Functional Testing | EFT-1 |
| Penetration Analysis | ---- |
| Covert Channel Analysis | ---- |

**Table 8. T1 Assurance Package**

| Assurance Component Summary | T1 |
|---|---|
| **Product Support** | ---- |
| **Evaluation Assurance Components** | |
| **Testing** | |
| Test Analysis | TA-1 |
| Independent Testing | IT-1 |
| **Review** | |
| Development Environment | ---- |
| Operational Support | ---- |
| **Analysis** | |
| Design | ---- |
| Implementation | ---- |

### Assurance Package T2

The T2 assurance package is intended to include IT products used within the DoD and Intelligence Community as well as the commercial world. For this assurance package a few additional development and evaluation assurance components, at their lowest levels, have been added. Some of the components from the previous package have been augmented but still remain on the low side of the component scale.

The intent of the development process assurance for this package, like the previous package, is only to establish that the external behavior of the product conforms to its user-level and administrative documentation without any analysis of the internal structure of the IT product's TCB. For this reason, only the claimed TCB protection properties, the formal models of these properties, the TCB interface description, and the TCB element list are required to enable functional testing. Support for TCB structuring is limited to process isolation and the separation of the protection critical TCB elements from the protection non-critical ones.

The intent of the operational support assurance for this package is to establish a minimal level of user and administrative guidance and product information that enables the correct product installation and use of the product's protection functions.

There are no requirements in this package pertaining to the environment in which the product is developed.

The intent of this package is to require the type of assurance evidence generated during the normal development process of a product that was rated B1 by TCSEC standards. Table 9 summarizes the generic assurance components that comprise the T2 assurance package.

**Table 9. T2 Assurance Package**

| Assurance Component Summary | T2 |
|---|---|
| **Development Assurance Components** | |
| **Development Process** | |
| **TCB Property Definition** | PD-2 |
| **TCB Design** | |
| TCB Element Identification | ID-2 |
| TCB Interface Definition | IF-1 |
| TCB Modular Decomposition | ---- |
| TCB Structuring Support | SP-1 |
| TCB Design Disciplines | ---- |
| **TCB Implementation Support** | ---- |
| **TCB Testing & Analysis** | |
| Functional Testing | FT-1 |
| Penetration Analysis | ---- |
| Covert Channel Analysis | ---- |
| **Operational Support** | |
| **User's Security Guidance** | UG-1 |
| **Administrative Guidance** | AG-1 |
| **Trusted Generation** | TG-1 |
| **Development Environment** | |
| **Life Cycle Definition** | ---- |
| **Configuration Management** | ---- |
| **Trusted Distribution** | ---- |
| **Development Evidence** | |
| **TCB Protection Properties** | EPP-2 |
| **Product Development** | EPD-1 |
| **Product Testing & Analysis** | |
| Functional Testing | EFT-1 |
| Penetration Analysis | ---- |
| Covert Channel Analysis | ---- |

**Table 9. T2 Assurance Package**

| Assurance Component Summary | T2 |
|---|---|
| **Product Support** | EPS-1 |
| **Evaluation Assurance Components** | |
| **Testing** | |
| Test Analysis | TA-1 |
| Independent Testing | IT-1 |
| **Review** | |
| Development Environment | ---- |
| Operational Support | OSR-1 |
| **Analysis** | |
| Design | ---- |
| Implementation | ---- |

### Assurance Package T3

The T3 assurance package is intended to include the highest level commercial IT products that incorporate protection functionality. Although most development assurance components are required at their lowest levels, the requirements of several development components are extended to capture (1) specific TCB properties, and (2) a rudimentary notion of support for product structure.

The intent of the development process assurance for this package is to establish that the external behavior of the product conforms to its user-level and administrative documentation without analysis of the internal structure of the IT product's TCB. Like the previous package, only the claimed TCB protection properties and their informal models, the TCB interface description, and the TCB element list are required to enable functional testing. Penetration testing is also enabled for this package using the same parameters. Support for TCB structuring is limited to process isolation and separation of the protection critical TCB elements from the protection non-critical ones.

The intent of the operational support assurance for this package is to establish a minimal level of user and administrative guidance and product information that enables the correct product installation and use of product protection functions.

The development environment assurances are intended to provide a minimal level of control over the product configuration and production. This level of development environment assurance is similar to that already present in most established commercial development organizations.

The intent of this package is to require the type of assurance evidence that is generated during the most stringent commercial development process. Table 10 summarizes the generic assurance components that comprise the T3 assurance package.

**Table 10. T3 Assurance Package**

| Assurance Component Summary | T3 |
|---|---|
| **Development Assurance Components** | |
| **Development Process** | |
| **TCB Property Definition** | PD-2 |
| **TCB Design** | |
| TCB Element Identification | ID-2 |
| TCB Interface Definition | IF-1 |
| TCB Modular Decomposition | ---- |
| TCB Structuring Support | SP-1 |
| TCB Design Disciplines | ---- |
| **TCB Implementation Support** | ---- |
| **TCB Testing & Analysis** | |
| Functional Testing | FT-1 |
| Penetration Analysis | PA-1 |
| Covert Channel Analysis | ---- |
| **Operational Support** | |
| **User's Security Guidance** | UG-1 |
| **Administrative Guidance** | AG-1 |
| **Trusted Generation** | TG-2 |
| **Development Environment** | |
| **Life Cycle Definition** | LC-1 |
| **Configuration Management** | CM-1 |
| **Trusted Distribution** | ---- |
| **Development Evidence** | |
| **TCB Protection Properties** | EPP-2 |
| **Product Development** | EPD-1 |
| **Product Testing & Analysis** | |
| Functional Testing | EFT-1 |
| Penetration Analysis | EPA-1 |

**Table 10. T3 Assurance Package**

| Assurance Component Summary | T3 |
|---|---|
| Covert Channel Analysis | ---- |
| **Product Support** | EPS-1 |
| **Evaluation Assurance Components** | |
| **Testing** | |
| Test Analysis | TA-2 |
| Independent Testing | IT-1 |
| **Review** | |
| Development Environment | DER-1 |
| Operational Support | OSR-1 |
| **Analysis** | |
| Design | DA-1 |
| Implementation | ---- |

**Assurance Package T4**

The T4 assurance package is intended to include IT products no longer of a commercial variety. This package includes several extensions to the assurance components of the previous two packages.

The intent of the development process assurance for this package is both to establish that the external behavior of the product conforms to its user-level and administrative documentation and to provide visibility into the internal structure of the IT product's TCB. For this reason, requirements for Descriptive Interface Specifications and modular decomposition have been added. The TCB element identification, functional testing, and penetration testing requirements have also been extended to support the added assurances of external behavior.

The intent of the operational support assurance for this package is to establish a level of user and administrative guidance and product information that enables the correct product installation and use of product protection functions. The developer is required to establish and document a policy for responding to consumer inquiries.

The development environment assurances are intended to provide a level of control over the product configuration and production, including well-defined coding standards and strict configuration management processes. This level of development environment assurance is more stringent than that used in most advanced commercial development organizations.

The intent of this package is to require more assurance evidence than that which is generated during commercial development oriented towards high-quality products. Table 11 summarizes the generic assurance components that comprise the T4 assurance package.

**Table 11. T4 Assurance Package**

| Assurance Component Summary | T4 |
|---|---|
| **Development Assurance Components** | |
| **Development Process** | |
| **TCB Property Definition** | PD-2 |
| **TCB Design** | |
| TCB Element Identification | ID-2 |
| TCB Interface Definition | IF-2 |
| TCB Modular Decomposition | MD-1 |
| TCB Structuring Support | SP-1 |
| TCB Design Disciplines | ---- |
| **TCB Implementation Support** | IM-1 |
| **TCB Testing & Analysis** | |
| Functional Testing | FT-2 |
| Penetration Analysis | PA-2 |
| Covert Channel Analysis | ---- |
| **Operational Support** | |
| **User's Security Guidance** | UG-1 |
| **Administrative Guidance** | AG-2 |
| **Trusted Generation** | TG-2 |
| **Development Environment** | |
| **Life Cycle Definition** | LC-2 |
| **Configuration Management** | CM-2 |

**Table 11. T4 Assurance Package**

| Assurance Component Summary | T4 |
|---|---|
| **Trusted Distribution** | --- |
| **Development Evidence** | |
| **TCB Protection Properties** | EPP-2 |
| **Product Development** | EPD-2 |
| **Product Testing & Analysis** | |
| Functional Testing | EFT-2 |
| Penetration Analysis | EPA-2 |
| Covert Channel Analysis | ---- |
| **Product Support** | EPS-2 |
| **Evaluation Assurance Components** | |
| **Testing** | |
| Test Analysis | TA-3 |
| Independent Testing | IT-2 |
| **Review** | |
| Development Environment | DER-2 |
| Operational Support | OSR-2 |
| **Analysis** | |
| Design | DA-2 |
| Implementation | CI-1 |

### Assurance Package T5

The T5 assurance package is the first of the high-assurance packages that are intended for environments where security is a primary operational consideration. These environments include, but are not restricted to, national defense, industrial process control, medical information processing, financial, and business controls.

The intent of the development process assurance for this package is to lead to IT products that are internally structured to the extent required by source-code analyses and by strict development environment requirements (e.g., strict configuration management). The source code analyses, which include penetration and storage-channel analyses, are intended to convey the evidence that the TCB protection properties are correctly implemented in the product.

The intent of the operational support assurance for this package is to establish a level of user and administrative guidance and product information that enables the correct product installation and use of product protection functions. The developer is required to establish and document a policy for responding to consumer inquiries.

The development environment assurances are intended to provide a well-defined level of control over the product configuration and production, including well-defined coding standards and strict configuration management processes. This level of development environment assurance is more stringent than that used in the most advanced commercial development organizations.

The intent of this package is to require the type of evidence that is necessary to assess whether this product is satisfactory for high assurance environments. This assurance evidence is the type generated during the normal development process of a product that was rated B2 by TCSEC standards. Evidence for the additional analyses is required. Table 12 summarizes the generic assurance components that comprise the T5 assurance package.

**Table 12. T5 Assurance Package**

| Assurance Component Summary | T5 |
|---|---|
| **Development Assurance Components** | |
| **Development Process** | |
| **TCB Property Definition** | PD-3 |
| **TCB Design** | |
| TCB Element Identification | ID-2 |
| TCB Interface Definition | IF-2 |
| TCB Modular Decomposition | MD-2 |
| TCB Structuring Support | SP-2 |
| TCB Design Disciplines | ---- |
| **TCB Implementation Support** | IM-3 |
| **TCB Testing & Analysis** | |
| Functional Testing | FT-3 |
| Penetration Analysis | PA-2 |
| Covert Channel Analysis | CCA-1 |
| **Operational Support** | |
| **User's Security Guidance** | UG-1 |
| **Administrative Guidance** | AG-2 |
| **Trusted Generation** | TG-2 |
| **Development Environment** | |

**Table 12. T5 Assurance Package**

| Assurance Component Summary | T5 |
|---|---|
| **Life Cycle Definition** | LC-2 |
| **Configuration Management** | CM-2 |
| **Trusted Distribution** | ---- |
| **Development Evidence** | |
| **TCB Protection Properties** | EPP-3 |
| **Product Development** | EPD-3 |
| **Product Testing & Analysis** | |
| Functional Testing | EFT-3 |
| Penetration Analysis | EPA-2 |
| Covert Channel Analysis | ECC-1 |
| **Product Support** | EPS-2 |
| **Evaluation Assurance Components** | |
| **Testing** | |
| Test Analysis | TA-4 |
| Independent Testing | IT-3 |
| **Review** | |
| Development Environment | DER-2 |
| Operational Support | OSR-2 |
| **Analysis** | |
| Design | DA-2 |
| Implementation | CI-1 |

### Assurance Package T6

The T6 assurance package is intended for environments where security is a major operational consideration. These environments include, but are not restricted to, national defense, industrial process control, medical information processing, financial, and business controls.

The intent of the development process assurance for this package is to lead to IT products that are internally structured to the highest possible extent required by source-code analyses and by strict development environment requirements (e.g., strict configuration management). The assurances included in this package include specific design disciplines that enable systematic code analysis (e.g., minimization of the TCB size and complexity). The source code analyses, which include systematic penetration and covert-channel analyses, are intended to convey evidence that the TCB protection properties are correctly implemented in the product.

The intent of the operational support assurance is to establish precise, specific administrative guidance on a per role basis that could mitigate or deter the exploitation of potential vulnerabilities.

The development environment assurances are intended to maintain a strict level of control over the product configuration and production, including demonstrable compliance with coding standards and strict configuration management processes. This level of development environment assurance is much more strict than that commonly used in the most advanced commercial development organizations.

The intent of this package is to require the type of evidence that is necessary to assess whether this product is satisfactory for high assurance environments. This assurance evidence is the type generated during the normal development process of a product that was rated B3 by TCSEC standards. Evidence for the additional analyses is required. Table 13 summarizes the generic assurance components that comprise the T6 assurance package.

**Table 13. T6 Assurance Package**

| Assurance Component Summary | T6 |
|---|---|
| **Development Assurance Components** | |
| **Development Process** | |
| **TCB Property Definition** | PD-3 |
| **TCB Design** | |
| TCB Element Identification | ID-2 |
| TCB Interface Definition | IF-2 |
| TCB Modular Decomposition | MD-3 |
| TCB Structuring Support | SP-3 |
| TCB Design Disciplines | DD-2 |
| **TCB Implementation Support** | IM-3 |
| **TCB Testing & Analysis** | |
| Functional Testing | FT-3 |
| Penetration Analysis | PA-2 |
| Covert Channel Analysis | CCA-2 |
| **Operational Support** | |
| **User's Security Guidance** | UG-1 |
| **Administrative Guidance** | AG-3 |
| **Trusted Generation** | TG-3 |
| **Development Environment** | |
| **Life Cycle Definition** | LC-3 |
| **Configuration Management** | CM-3 |

**Table 13. T6 Assurance Package**

| Assurance Component Summary | T6 |
|---|---|
| **Trusted Distribution** | ---- |
| **Development Evidence** | |
| **TCB Protection Properties** | EPP-3 |
| **Product Development** | EPD-4 |
| **Product Testing & Analysis** | |
| Functional Testing | EFT-3 |
| Penetration Analysis | EPA-2 |
| Covert Channel Analysis | ECC-2 |
| **Product Support** | EPS-3 |
| **Evaluation Assurance Components** | |
| **Testing** | |
| Test Analysis | TA-4 |
| Independent Testing | IT-3 |
| **Review** | |
| Development Environment | DER-3 |
| Operational Support | OSR-3 |
| **Analysis** | |
| Design | DA-3 |
| Implementation | CI-3 |

### Assurance Package T7

The T7 assurance package is intended for environments where security is the major operational consideration. These environments include, but are not restricted to, national defense, industrial process control, life-support systems, as well as special aeronautical and space applications. Products developed at this level of assurance are expected to include special development environments, not just special development processes.

The intent of the development process assurance for this package is to lead to IT products that are internally structured to the highest possible extent required by formal design and source-code analyses and by very strict development environment requirements (e.g., automated configuration management and configuration management safeguards). The assurances included in this package include the state-of-the-art design disciplines that enable formal analysis and systematic code analysis (e.g., minimization of the TCB size and complexity). The source code analyses, which include verification of penetration-resistance and covert-channel analysis, are intended to convey evidence that the TCB protection properties are correctly designed and implemented in the product.

The intent of the operational support assurance is to establish precise, specific administrative guidance on a per role basis that could mitigate or deter the exploitation of potential vulnerabilities.

The development environment assurances are intended to maintain the highest level of control over the product configuration and production, including demonstrable compliance with coding standards and strict configuration management processes. This level of development environment assurance is the most stringent used for any IT product production.

The intent of this package is to require the type of evidence that is necessary to assess whether the IT product is satisfactory for the highest levels of assurance. This assurance evidence is the type generated during the normal development process of a product that was rated A1 by TCSEC standards. Evidence of the formal analyses becomes necessary. Table 14 summarizes the generic assurance components that comprise the T7 assurance package.

**Table 14. T7 Assurance Package**

| Assurance Component Summary | T7 |
|---|---|
| **Development Assurance Components** | |
| **Development Process** | |
| **TCB Property Definition** | PD-4 |
| **TCB Design** | |
| TCB Element Identification | ID-2 |
| TCB Interface Definition | IF-3 |
| TCB Modular Decomposition | MD-3 |
| TCB Structuring Support | SP-3 |
| TCB Design Disciplines | DD-2 |
| **TCB Implementation Support** | IM-4 |
| **TCB Testing & Analysis** | |
| Functional Testing | FT-3 |
| Penetration Analysis | PA-2 |
| Covert Channel Analysis | CCA-3 |
| **Operational Support** | |
| **User's Security Guidance** | UG-1 |
| **Administrative Guidance** | AG-3 |
| **Trusted Generation** | TG-3 |
| **Development Environment** | |
| **Life Cycle Definition** | LC-3 |
| **Configuration Management** | CM-4 |
| **Trusted Distribution** | TD-1 |

## Table 14. T7 Assurance Package

| Assurance Component Summary | T7 |
|---|---|
| **Development Evidence** | |
| **TCB Protection Properties** | EPP-4 |
| **Product Development** | EPD-5 |
| **Product Testing & Analysis** | |
| Functional Testing | EFT-3 |
| Penetration Analysis | EPA-2 |
| Covert Channel Analysis | ECC-2 |
| **Product Support** | EPS-3 |
| **Evaluation Assurance Components** | |
| **Testing** | |
| Test Analysis | TA-5 |
| Independent Testing | IT-4 |
| **Review** | |
| Development Environment | DER-3 |
| Operational Support | OSR-3 |
| **Analysis** | |
| Design | DA-3 |
| Implementation | CI-3 |

**Table 15. Assurance Packages Summary**

| Assurance Components | T1 | T2 | T3 | T4 | T5 | T6 | T7 |
|---|---|---|---|---|---|---|---|
| **Development Assurance Components** | | | | | | | |
| **Development Process** | | | | | | | |
| **TCB Property Definition** | PD-1 | PD-2 | PD-2 | PD-3 | PD-3 | PD-3 | PD-4 |
| **TCB Design** | | | | | | | |
| TCB Element Identification | ID-1 | ID-2 | ID-2 | ID-2 | ID-2 | ID-2 | ID-2 |
| TCB Interface Definition | IF-1 | IF-1 | IF-1 | IF-2 | IF-2 | IF-2 | IF-3 |
| TCB Modular Decomposition | ---- | ---- | ---- | MD-1 | MD-2 | MD-3 | MD-3 |
| TCB Structuring Support | ---- | SP-1 | SP-1 | SP-1 | SP-2 | SP-3 | SP-3 |
| TCB Design Disciplines | ---- | ---- | ---- | ---- | ---- | DD-2 | DD-2 |
| **TCB Implementation Support** | ---- | ---- | ---- | IM-1 | IM-3 | IM-3 | IM-4 |
| **TCB Testing & Analysis** | | | | | | | |
| Functional Testing | FT-1 | FT-1 | FT-1 | FT-3 | FT-3 | FT-3 | FT-3 |
| Penetration Analysis | ---- | ---- | PA-1 | PA-2 | PA-2 | PA-2 | PA-2 |
| Covert Channel Analysis | ---- | ---- | ---- | ---- | CCA-1 | CCA-2 | CCA-3 |
| **Operational Support** | | | | | | | |
| **User's Security Guidance** | UG-1 | UG-1 | UG-1 | UG-1 | UG-1 | UG-1 | UG-1 |
| **Administrative Guidance** | AG-1 | AG-1 | AG-2 | AG-2 | AG-2 | AG-3 | AG-3 |
| **Trusted Generation** | ---- | TG-1 | TG-2 | TG-2 | TG-2 | TG-3 | TG-3 |
| **Development Environment** | | | | | | | |
| **Life Cycle Definition** | ---- | ---- | LC-1 | LC-2 | LC-2 | LC-3 | LC-3 |
| **Configuration Management** | ---- | ---- | CM-1 | CM-2 | CM-2 | CM-3 | CM-4 |
| **Trusted Distribution** | ---- | ---- | ---- | ---- | ---- | ---- | TD-1 |
| **Development Evidence** | | | | | | | |
| **TCB Protection Properties** | EPP-1 | EPP-2 | EPP-2 | EPP-3 | EPP-3 | EPP-3 | EPP-4 |
| **Product Development** | EPD-1 | EPD-1 | EPD-1 | EPD-3 | EPD-3 | EPD-4 | EPD-5 |
| **Product Testing & Analysis** | | | | | | | |
| Functional Testing | EFT-1 | EFT-1 | EFT-1 | EFT-3 | EFT-3 | EFT-3 | EFT-3 |
| Penetration Analysis | ---- | ---- | EPA-1 | EPA-2 | EPA-2 | EPA-2 | EPA-2 |
| Covert Channel Analysis | ---- | ---- | ---- | ---- | ECC-1 | ECC-2 | ECC-2 |
| **Product Support** | ---- | EPS-1 | EPS-1 | EPS-2 | EPS-2 | EPS-3 | EPS-3 |

## Table 15. Assurance Packages Summary

| Assurance Components | T1 | T2 | T3 | T4 | T5 | T6 | T7 |
|---|---|---|---|---|---|---|---|
| **Evaluation Assurance Components** | | | | | | | |
| **Testing** | | | | | | | |
| Test Analysis | TA-1 | TA-1 | TA-2 | TA-3 | TA-4 | TA-4 | TA-5 |
| Independent Testing | IT-1 | IT-1 | IT-1 | IT-2 | IT-3 | IT-3 | IT-4 |
| **Review** | | | | | | | |
| Development Environment | ---- | ---- | DER-1 | DER-2 | DER-2 | DER-3 | DER-3 |
| Operational Support | ---- | OSR-1 | OSR-1 | OSR-2 | OSR-2 | OSR-3 | OSR-3 |
| **Analysis** | | | | | | | |
| Protection Properties | ---- | ---- | ---- | ---- | ---- | ---- | ---- |
| Design | ---- | ---- | DA-1 | DA-2 | DA-2 | DA-3 | DA-3 |
| Implementation | ---- | ---- | ---- | CI-1 | CI-2 | CI-3 | CI-3 |