



GFI LANguard N.S.S. v8 - New version out now!

Voted Favorite Commercial Security Tool for two consecutive years by NMAP users.

Download free 30-day trial today! - www.gfi.com/lannetscan/

NCSC-TG-010: Aqua book

A Guide to Understanding Security Modeling in Trusted Systems

- Published: **Oct 16, 2002**
- Updated: **Oct 16, 2002**
- Section: [Network Security Library :: NCSC&DoD Rainbow series](#)
- Author: [The Editor](#)
- Company: [WindowSecurity.com](#)
- Rating: **4/5 - 3 Votes**



FOREWORD

The National Computer Security Center is issuing A Guide to Understanding Security Modeling in Trusted Systems as part of the "Rainbow Series" of documents produced by our Technical Guidelines Program. In the Rainbow Series, we discuss, in detail, the features of the Department of Defense Trusted Computer System Evaluation Criteria (DoD 5200.28-STD) and provide guidance for meeting each requirement. The National Computer Security Center, through its Trusted Product Evaluation Program, evaluates the security features and assurances of commercially-produced computer systems. Together, these programs ensure that organizations are capable of protecting their important data with trusted computer systems. Security modeling, in its various forms, is an important component of the assurance of the trust of a computer system.

A Guide to Understanding Security Modeling in Trusted Systems is intended for use by personnel responsible for developing models of the security policy of a trusted computer system. At lower levels of trust, this model is generally the system's philosophy of protection. At higher trust levels, this also includes informal and formal models of the protection mechanisms within a system. This guideline provides information on many aspects of security modeling, including the process of developing a security policy model, security modeling techniques, and specific ways to meet the requirements of the Department of Defense Trusted Computer System Evaluation Criteria.

As the Director, National Computer Security Center, I invite your suggestions for revising this document. We plan to review and revise this document as the need arises.

Patrick R. Gallagher, Jr. October 1992

Director

National Computer Security Center

ACKNOWLEDGMENTS

Special recognition and acknowledgment for their contributions to this document are extended to the following: Dr. James Williams, as primary author of this document; Barbara Mayer, Capt. James Goldston, USAF, and Lynne Ambuel, for project management.

Special thanks are also given to the many evaluators, vendors, and researchers whose careful reviews and suggestions contributed to this document, with special recognition for the advice of David Bell, Daniel Schnackenberg, Timothy Levin, Marshall Abrams, Deborah Bodeau, Leonard La Padula, Jonathan Millen, William L. Harkness, Thomas A. Ambrosi, Paul Pittelli, Dr. Michael Sinutko, Jr, and COL Rayford Vaughn, USA.

TABLE OF CONTENTS

FOREWORD	i
ACKNOWLEDGMENT	ii
1. INTRODUCTION	1
1.1 Background	1
1.2 Purpose	2
1.3 Control Objectives	3
1.3.1 The Assurance Objective	3
1.3.2 The Security Policy Objective	4
1.4 Historical Overview	5
1.5 Content and Organization	7
2. OVERVIEW OF A SECURITY MODELING PROCESS	9
2.1 Security Models and Their Purpose	9
2.2 Security Modeling in the System Development Process	11
2.3 Identifying the Contents of a Security Model	14
2.3.1 Step 1: Identify Requirements on the External Interface	16
2.3.2 Step 2: Identify Internal Requirements	18
2.3.3 Step 3: Design Rules of Operation for Policy Enforcement	20
2.3.4 Step 4: Determine What is Already Known	21
2.3.5 Step 5: Demonstrate Consistency and Correctness	22
2.3.6 Step 6: Demonstrate Relevance	23
2.4 Presentation for Evaluation and Use	24
3 .SECURITY MODELING TECHNIQUES	27

3.1 Basic Concepts	27
3.1.1 Data Structures and Storage Objects	28
3.1.2 Processes and Subjects	29
3.1.3 Users and User Roles	31
3.1.4 I/O Devices	33
3.1.5 Security Attributes	35
3.1.6 Partially Ordered Security Attributes	37
3.1.7 Nondisclosure Levels	39
3.1.8 Unlabeled Entities and the Trusted Computing Base	40
3.2 Nondisclosure and Mandatory Access Control	41
3.2.1 External-Interface Requirements and Model	43
3.2.2 Information-Flow Model	45
3.2.3 Application to the Reference Monitor Interface	47
3.2.4 Access-Constraint Model	48
3.2.5 Tailoring the Models	51
3.3 Need-to-Know and Discretionary Access Control	53
3.3.1 DAC Requirements and Mechanisms	54
3.3.2 User Groups and User Roles	55
3.3.3 Sources of Complexity and Weakness	56
3.3.4 Tight Per-User Access Control	59
3.4 TCB Subjects-Privileges and Responsibilities	61
3.4.1 Identifying Privileges and Exemptions	62
3.4.2 Responsible Use of Exemptions	65
3.5 Integrity Modeling	66
3.5.1 Objectives, Policies, and Models	67
3.5.2 Error Detection and Recovery	68
3.5.3 Encapsulation and Level-Based Access Control	71
4. TECHNIQUES FOR SPECIFIC KINDS OF SYSTEM	5
4.1 Operating Systems	75
4.1.1 Traditional Access Control Models	75

4.1.2 The Models of Bell and La Padula	77	
4.2 Networks and Other Distributed Systems		78
4.2.1 Systems and Their Components	78	
4.2.2 Model Structure and Content	80	
4.2.3 Network Security Models	82	
4.2.4 Individual Component Security Models		84
4.2.5 Underlying Models of Computation	85	
4.3 Database Management Systems	86	
4.3.1 System Structure	87	
4.3.2 Integrity Mechanisms and Policies	89	
4.3.3 Aggregation	90	
4.3.4 Inference	91	
4.3.5 Partially Automated Labeling	92	
4.4 Systems with Extended Support for Label Accuracy		94
4.4.1 Factors Inhibiting Accuracy in Labeling	94	
4.4.2 Floating Sensitivity Labels	95	
4.4.3 Compartmented Mode Workstations (CMW)	95	
4.4.4 The Chinese Wall Security Policy	96	
5. MEETING THE REQUIREMENTS	99	
5.1 Stated Requirements on the Security Model	99	
5.2 Discussion of the B1 Requirements	100	
5.3 Discussion of the B2 Requirements	103	
5.4 Discussion of the B3 Requirements	104	
5.5 Discussion of the A1 Requirements	105	
APPENDIX A. SECURITY LEVELS AND PARTIALLY ORDERED SETS		107
A.1 Terminology	107	
A.2 Embeddings	109	
A.3 Cartesian Products	109	
A.4 Duality	110	
APPENDIX B. AVAILABLE SUPPORT TOOLS		113
B.1 FDM: the Formal Development Methodology	114	
B.2 GVE: the Gypsy Verification Environment	115	
APPENDIX C. PHILOSOPHY OF PROTECTION OUTLINE		17
APPENDIX D. SECURITY MODEL OUTLINE		21
APPENDIX E. GLOSSARY	127	
REFERENCES	45	
1. INTRODUCTION		

This document provides guidance on the construction, evaluation, and use of security policy models for automated information systems (AIS) used to protect sensitive information whose unauthorized disclosure, alteration, loss, or destruction must be prevented. In this context, sensitive information includes classified information as well as unclassified sensitive information.

1.1 BACKGROUND

The National Computer Security Center (NCSC) was established in 1981 and acquired its present name in 1985. Its main goal is to encourage the widespread availability of trusted AIs. In support of this goal, the DoD Trusted Computer System Evaluation Criteria (TCSEC) was written in 1983. It has been adopted, with minor changes, as a DoD standard for the protection of sensitive information in DoD computing systems. [NCSC85] The TCSEC is routinely used for the evaluation of commercial computing products prior to their accreditation for use in particular environments. This evaluation process is discussed in Trusted Product Evaluations: A Guide for Vendors. [NCSC90c]

The TCSEC divides AIs into four main divisions, labeled D, C, B, and A, in order of increasing security protection and assurance. Divisions C through A are further divided into ordered subdivisions referred to as classes. For all classes (C1, C2, B1, B2, B3 and A1), the TCSEC requires system documentation that includes a philosophy of protection. For classes B1, B2, B3, and A1, it also requires an informal or formal security policy model.

Although the TCSEC is oriented primarily toward operating systems, its underlying concepts have been applied much more generally. In recognition of this, the NCSC has published a Trusted Network Interpretation [NCSC87] and a Trusted Database Management System Interpretation. [NCSC91] In addition, the NCSC also provides a series of guidelines addressing specific TCSEC requirements, of which this document is an example.

1.2 PURPOSE

This guideline is intended to give vendors and evaluators of trusted systems a solid understanding of the modeling requirements of the TCSEC and the Trusted Network Interpretation of the TCSEC (TNI). It presents modeling and philosophy of protection requirements for each of the classes in the TCSEC, describes possible approaches to modeling common security requirements in various kinds of systems, and explains what kinds of models are useful in an evaluation. It is intended for vendors, evaluators, and other potential builders and users of security models.

This guideline discusses the philosophy of protection requirement, explains how it relates to security modeling, and provides guidance on documentation relating to the system's philosophy of protection and security policy model. It explains the distinction between informal and formal security policy models as well as the relationships among application-independent models, application-dependent security models, and model interpretations. It also explains which techniques may be used to meet the modeling requirements at levels B1 through A1 as well as the advantages and disadvantages of the various modeling techniques. Finally, it discusses the specific TCSEC modeling requirements.

This guideline also addresses human aspects of modeling. It describes how modeling captures basic security requirements and how the security modeling effort leads to better systems and provides a basis for increased assurance of security.

Finally, this guideline answers common questions about NCSC recommendations regarding

the construction of security models. Security policies and models are supplied by vendors in response to customer needs and TCSEC requirements; they are not supplied by the NCSC or the TCSEC. The TCSEC does, however, set minimum requirements in the areas of mandatory and discretionary access control. The TCSEC does not require particular implementation techniques; this freedom applies, by extension, to modeling techniques. Acceptability of a technique depends on the results achieved. More specifically, a security model must provide a clear and accurate description of the security policy requirements, as well as key ideas associated with their enforcement. Moreover, one must be able to validate the model using assurance techniques appropriate to the proposed evaluation class. Any vendor-supplied modeling technique which appears to be compatible with the security modeling requirements will be seriously considered during the course of a product evaluation.

Topics which are closely related to security modeling include formulation of security policy objectives, design specification and verification, covert channel analysis, and implementation correspondence analysis. These topics are addressed only to the extent necessary to establish their influence on the security modeling process. All but the first of these topics are addressed in other guidelines in this series. Security objectives for trusted systems traditionally include nondisclosure, integrity, and denial of service. [cf NCSC88, AIS Security] However, the modeling of denial of service is not addressed in this guideline, due to the lack of adequate literature on this topic. This guideline is written with the understanding that security modeling will continue to evolve in response to new policy objectives and modeling techniques associated with future trusted system designs.

Reading and understanding this guideline is not sufficient to allow an inexperienced individual to develop a model. Rather, he or she should read this document in conjunction with one or more of the published models in the list of references. This guideline assumes a general familiarity with the TCSEC and with computer security. A general text such as Building a Secure Computer System [GASS87] may provide useful background reading. Additional mathematical background needed for formal modeling can be found in general texts on mathematical structures such as Introduction to Mathematical Structures [GAL089].

The approaches to security modeling presented in this document are not the only possible approaches to satisfying TCSEC modeling requirements, but are merely suggested approaches. The presentation of examples illustrating these approaches does not imply NCSC endorsement of the products on which these examples are based. Recommendations made in this document are not supplementary requirements to the TCSEC. The TCSEC itself (as supplemented by announced interpretations [NCSC87, NCSC88a, NCSC88b, NCSC91]) is the only metric used by the NCSC to evaluate trusted computing products.

1.3 CONTROL OBJECTIVES

The requirements of the TCSEC were derived from three main control objectives: assurance, security policy, and accountability. The security modeling requirements of the TCSEC support the assurance and security policy control objectives in systems rated B1 and above.

1.3.1 THE ASSURANCE OBJECTIVE

The TCSEC assurance control objective states, in part,

"Systems that are used to process or handle classified or other sensitive information must be designed to guarantee correct and accurate interpretation of the security policy and must not distort the intent of that policy." [NCSC85, § 5.3.3]

Assurance in this sense refers primarily to technical assurance rather than social assurance. It

involves reducing the likelihood that security mechanisms will be subverted as opposed to just improving people's confidence in the utility of the security mechanisms.

1.3.2 THE SECURITY POLICY OBJECTIVE

The TCSEC security policy control objective states, in part,

"A statement of intent with regard to control over access to and dissemination of information, to be known as the security policy, must be precisely defined and implemented for each system that is used to process sensitive information. The security policy must accurately reflect the laws, regulations, and general policies from which it is derived." [NCSC85, § 5.3.1]

The security policy objective given in the TCSEC covers "mandatory security policy," "discretionary security policy," and "marking" objectives. The mandatory security policy objective requires the formulation of a mandatory access control (MAC) policy that regulates access by comparing an individual's clearance or authorization for information to the classification or sensitivity designation of the information to be accessed. The discretionary access control objective requires the formulation of a discretionary access control (DAC) policy that regulates access on the basis of each individual's need-to-know. Finally, the marking objective gives labeling requirements for information stored in and exported from systems designed to enforce a mandatory security policy.

The access controls associated with security policies serve to enforce nondisclosure and integrity. Nondisclosure controls prevent inappropriate dissemination of information. Integrity controls prevent inappropriate modification of information.

The security policy control objective requires that the security policy accurately reflect the laws, regulations, and general policies from which it is derived. These may include the revised DoD Directive 5200.28, Security Requirements for Automated Information Systems (AISs). [DOD88a] Section D of Directive 5200.28 gives policy relating to both integrity and nondisclosure. It mentions safeguards "against sabotage, tampering, denial of service, espionage, fraud, misappropriation, misuse, or release to unauthorized users." Enclosure 2 defines relevant terms and, in particular, extends the concept of "user" to include processes and devices interacting with an automated information system on behalf of users. Enclosure 3 gives general security requirements, including data-integrity requirements. The revised 5200.28 does not use the MAC/DAC terminology. Instead, it requires accurate marking of sensitive information and the existence of an (undifferentiated) access control policy based partly on the identity of individual users. It treats need-to-know as a form of least privilege.

Security policy relating to nondisclosure stems from Executive Order 12356 [REAG82] and, in the case of DoD systems, from DoD 5200.1-R. [DOD86] Depending on the application, the security policy may be constrained by a variety of other regulations as well. The collection, maintenance, use, and dissemination of personal information is protected by DoD Directive 5400.11 [DOD82, § E.2] and by Public Law 100-503 [CONG88]. Classified and other sensitive information needed for the conduct of federal programs is protected by the Computer Security Act of 1987, Public Law 100-235 [CONG87], which requires safeguards against loss and unauthorized modification.

1.4 HISTORICAL OVERVIEW

The starting point in modeling a MAC policy is the observation that security levels are partially ordered (see Appendix A). This fact was first reflected in the design of the ADEPT-50,

an IBM 360-based, time-sharing system with both mandatory and discretionary controls. [WEIS69, LAND81] The central role of partial orderings in MAC policy models was then recognized in subsequent work. [POPE73; BELL73; BELL74a] These partial orderings on security levels have become known as dominance relations.

Independently of this, access control matrices were used by Lampson and Graham [LAMP71; GRAH72] to represent protection data. These works modeled discretionary access control using matrices whose rows and columns represented subjects and objects. Subjects were active entities such as processes and users. Objects were entities such as data structures that played a passive role, but often included subjects as well. Each entry in an access matrix told what a given subject was allowed to do with a given object by giving a set of allowed operations such as read, write, execute, or change ownership. The AIS used the access matrix to mediate all accesses by subjects to objects.

An Air Force Computer Security Technology Planning Study [ANDE72], referred to as "the Anderson Report," discussed use of a reference monitor to enforce nondisclosure policies and advanced the idea of formal security verification- the use of rigorous mathematical proofs to give (partial) assurance that a program design satisfies stated security requirements. This idea was investigated by Schell, Downey and Popek. [SCHE73]

At that time, Bell and La Padula adapted access control matrices for use in modeling both mandatory and discretionary access controls and codified the reference monitor concept using state machines. Their machine states included access matrices and other security-relevant information and are now often referred to as protection states. Having established the necessary framework, they observed that a portion of DoD policy for nondisclosure of classified information could be formalized as invariant properties of protection states [LAPA73], that is, as properties which hold in all reachable states of the system. Their invariants constrained the allowed accesses between subjects and objects. The most significant of these was their *-property It included a form of the simple security property and guaranteed that the security level of every object read by an untrusted subject was at or below the security level of every object written by that subject. [BELL74] These ideas and their use in enforcing nondisclosure are covered in detail in Sections 3.2.4 and 4.1.

To complete their state machine, Bell and La Padula introduced a set of state transformations, called rules of operation, that modeled basic changes in a protection state and then rigorously proved that the rules of operation preserved the identified state invariants. [LAPA73, BELL74, BELL76] This work contains the first widely discussed use of mathematical proof for studying multilevel security.

In parallel with Bell and La Padula, Walter and others developed a similar approach to verifying multilevel security. [WALT74, WALT74a] A strong point of the work by Walter et al. is the use of modularity and data abstraction techniques that have since been accepted as indispensable for verifying large systems.

The intent of the state invariants identified by Bell and La Padula is that information is allowed to flow from one entity to another only if the second entity is at an equal or higher security level than the first. Denning and others attempted to formalize this idea directly using "information flow" models. [DENN76, DENN77, COHE77, REIT79, ANDR80]. These models differed in style from access control models in that they referred explicitly to information flow, but, in

contrast to more recent investigations, did not actually define information flow. Denning's work did, however, point out an interesting distinction. In the conditional assignment

```
if a = 0 then b := c,
```

information flows explicitly from c to b (when a = 0) and implicitly from a to b (when b = c). Denning also pointed out that, in the case of the ADEPT-50, access control can easily allow implicit information flow. [DENN77] This problem is discussed in detail in Section 3.2.

Discrepancies between information flow and access control open up the possibility of covert channels- paths of information flow that malicious processes can use to bypass the access control mechanism and thereby violate underlying security objectives. [cf LAMP73] Information flow models and concern over covert channels have led to the development of techniques and tools for covert channel analysis that are capable of detecting a variety of covert channels. [MILL76, MILL81, FEIE77, FEIE80]

The developments just described provided the technical background for the security modeling requirements given in the TCSEC. A variety of important developments not explicitly reflected in the TCSEC have taken place in the last decade and will be presented later in this guideline. The class of security policies that can be formally modeled has been greatly extended, beginning with Biba's (pre-TCSEC) work on integrity. [BlBA77] Work has also gone into tailoring nondisclosure security to particular applications. Finally, various external-interface models have been developed that do not constrain internal system structure, an example of which is the noninterference model of Goguen and Meseguer. [GOGU82] These models provide a rationale for rigorously assessing new approaches to access control.

1.5 CONTENT AND ORGANIZATION

Section 2 presents an overview of the security modeling process, with emphasis on correctness and utility. Section 3 presents technical detail on how to model concepts of interest, including nondisclosure and integrity policies, mandatory and discretionary access controls, and exemption from access control within the Trusted Computing Base (TCB).

Section 4 shows how to apply the techniques from Section 3 to various kinds of systems; including operating systems, networks, database systems, and multilevel workstations. Finally, Section 5 summarizes all of the TCSEC security modeling requirements for B1, B2, B3, and A1 computing systems.

Appendix A presents facts about lattices and partially ordered sets that are needed for Sections 3 and 4. Appendix B contains brief descriptions of available support tools. Appendices C and D contain suggested outlines for a philosophy of protection and a security policy model. Finally, Appendix E is a glossary giving definitions of technical terms. This glossary includes all terms that are introduced in italics throughout the guideline.

When TCSEC requirements are discussed in this guideline, they are identified either by the key words "must" or "shall" or by explicit quotations from the TCSEC. By way of contrast, when desirable but optional actions and approaches are discussed, they are presented without exhortation and are instead accompanied by an explanation of specific advantages or benefits. In a few cases, possible requirements are designated by "should," because the implications of the TCSEC are not fully understood or agreed upon.

2. OVERVIEW OF A SECURITY MODELING PROCESS

A security model precisely describes important aspects of security and their relationship to system behavior. The primary purpose of a security model is to provide the necessary level of understanding for a successful implementation of key security requirements. The security policy plays a primary role in determining the content of the security model. Therefore, the successful development of a good security model requires a clear, well-rounded security policy. In the case of a formal model, the development of the model also must rely on appropriate mathematical techniques of description and analysis for its form.

Sections 2.1 and 2.2 explain what security models describe, why they are useful, and how they are used in the design of secure systems. Section 2.3 introduces general definitions relating to security models and explains how security models are created. Finally, Section 2.4 discusses the presentation of a security model in a modeling document.

2.1 SECURITY MODELS AND THEIR PURPOSE

Early security models focused primarily on nondisclosure of information. More recently, the importance of data as a basis for decisions and actions has stimulated interest in integrity models.

[DOD88a, WHIT84] For example, nondisclosure properties alone do not protect against viruses that can cause unauthorized, malicious modification of user programs and data.

A wide variety of concepts can impact nondisclosure and integrity in particular system designs. As a result, the content of security models is quite varied. Their primary purpose is to provide a clear understanding of a system's security requirements. Without such an understanding, even the most careful application of the best engineering practices is inadequate for the successful construction of secure systems.

Inadequacies in a system can result either from a failure to understand requirements or from flaws in their implementation. The former problem, defining what a system should do, is relatively difficult in the case of security. The definition must be precise in order to prevent undesired results, or subtle flaws, during the implementation of the system design.

During the entire design, coding, and review process, the modeled security requirements may be used as precise design guidance, thereby providing increased assurance that the modeled requirements are satisfied by the system. The precision of the model and resulting guidance can be significantly improved by casting the model in a formal language.

Once the system has been built, the security model serves the evaluation and accreditation processes. It contributes to the evaluators' judgement of how well the developers have understood the security policy being implemented and whether there are inconsistencies between security requirements and system design. Moreover, the security model provides a mechanism for tailoring the evaluation process to the vendor's needs because it presents the security concept that is supported by the system being evaluated. The inclusion of particular facts in the security model proclaims to evaluators and potential customers that those facts are validated at the level of assurance which the TCSEC associates with that system's evaluation class.

Upon successful evaluation and use, the security model provides a basis for explaining security-relevant aspects of system functionality. Later, during maintenance, it provides a basis for

guidance in making security-relevant modifications. Finally, by suppressing inessential design detail, security models facilitate a broader understanding of security that can be applied to increasingly larger classes of systems. Among the many examples of such generalizations is the adaptation of traditional reference monitor concepts referenced in the TNI to provide a basis for understanding network security requirements. [NCSC87]

The intended purpose of a security model suggests several desirable properties. The requirements captured by a good model pertain primarily to security, so that they do not unduly constrain the functions of the system or its implementation. A good model accurately represents the security policy that is actually enforced by the system. Thus, it clarifies both the strengths and the potential limitations of the policy. (As an extreme example, if the system can simply declassify all objects and then proceed normally, as in McLean's System Z [MCLE87], a good model would show this.) Finally, a good model is simple and therefore easy to understand; it can be read and fully understood in its entirety by its intended audience. This last property cannot be achieved without significant care in choosing the contents, organization, and presentation of the security model. For example, the desire to provide a security model with the "look and feel" of UNIX, might need to be tempered by the need for simplicity and abstraction. [cf NCSC90b, Sec. 6.2]

2.2 SECURITY MODELING IN THE SYSTEM DEVELOPMENT PROCESS

Security requirements are best identified early in the system development process. Not identifying security requirements in a timely fashion is likely to have devastating effects on security assurance, security and application functionality, development schedule, and overall development costs. For example, in the case of a development using DOD-STD-2167A, [DOD88] this identification process would be part of the system requirements analysis. The identification of security requirements begins with the identification of high-level security objectives (as described in Section 1.3) and the methods by which they are to be met, including automated, procedural, and physical protection methods. This identification of security requirements and their derivation from identified higher-level security objectives is the initial material for a philosophy of protection (POP). As indicated in Appendix C, the philosophy of protection may also include a broad range of other topics such as the structure of the trusted computing base (TCB) and physical and procedural security mechanisms.

Those requirements in the philosophy of protection which deal with automated protection methods provide an initial definition of security for a security policy model. The model's purpose is to precisely state these requirements and to compare them with key aspects of the security enforcement mechanism. A security policy model in this sense contains two essential portions: a "definition of security" portion that captures key security requirements and a "rules of operation" portion that shows how the definition of security is enforced.

The model's definition of security can be used to avoid major system development errors. It can be used to guide the design of basic software protection mechanisms and to influence the design, selection and use of underlying firmware and hardware protection mechanisms. The initial draft model, and supporting documentation, provides guidance as to system security during reviews of the system design. However, there often are discrepancies between the design and the model. Some of these are resolvable and can be identified and corrected during the normal design review process. In some cases, however, discrepancies are unavoidable and can only be resolved by making some assumptions that simplify the problem. These assumptions need to be justifiable,

based on the model. These discrepancies can also be addressed through procedural restrictions on the use of the system. There are some portions of a security model that may require design information that is not initially available and must, therefore, be postponed. Possible examples include detailed rules of operation for a security kernel and security models for specific security-critical processes. In such cases, the system designer must ensure that discrepancies are noted and the completed system will satisfy the completed model.

To ensure that a design satisfies modeled security requirements, it is necessary to give a model interpretation which shows how the model relates to the system. For evaluation classes B1 and B2, this involves explaining how each concept in the model is embodied by the system design and informally demonstrating that the modeled requirements are met. Since the model's rules of operation must conform to the model's definition of security, the model interpretation need demonstrate only that the rules of operation are adhered to. For classes B3 and A1, the model interpretation is done in two steps. The design, as reflected in a top-level specification (TLS), is shown to be consistent with the model, and the implementation is shown to be consistent with the TLS. For Class B3, an informal descriptive top-level specification (DTLS) is used, an informal correspondence from the DTLS to the model is performed, and the implementation is shown to be consistent with the DTLS. At Class A1, the DTLS is supplemented with a formal top-level specification (FTLS), a formal verification proves that the FTLS is consistent with the model, and the implementation is shown to be consistent with the FTLS. A fuller summary of model-interpretation requirements is given in Section 5.

The role of security modeling in relation to other aspects of system development is summarized in Figure 2.1. Aspects that do not directly involve modeling, per se, are shaded in grey.

Requirements concerning the model are summarized at the beginning of Section 5. The broadest document is the philosophy of protection (POP); it covers higher-level security objectives, derived security policies constraining the design and use of the system, and the protection mechanisms enforced by the TCB. The POP, the security policy, and the model all cover the system's definition of security. Both the POP and the model cover key aspects of the TCB protection mechanisms. At B2 and above, the formal model supports a rigorous proof showing that the rules of operation enforce the definition of security, and the DTLS gives a functional description of the TCB protection mechanisms with emphasis on the TCB interface. At A1, the FTLS formalizes a large portion of the DTLS in order to verify that the TCB design satisfies the modeled security requirements.

Figure 2.1. Security Documentation

The above paragraphs refer to a single security model but networks, database systems, and other complex systems may involve several security models or submodels. When a system is made up of several complex components or subsystems, interfaces between the components or subsystem layers must be modeled, if they play a key role in security protection. In this case, the best approach may be to develop separate models for each component or layer in order to show how the various subsystems contribute to overall system security. A danger with this approach, however, is that the combined effect of the various submodels may not be obvious. This is discussed further in Section 4.

2.3 IDENTIFYING THE CONTENTS OF A SECURITY MODEL

The most basic strategy in identifying the contents of a security model is perhaps that of divide

and conquer. The modeling effort may be subdivided according to higher-level security objectives.

Requirements on a system can be mapped to derived requirements on subsystems. Requirements for a particular objective can be classified according to level in a requirements hierarchy.

Security modeling provides a five-stage elaboration hierarchy for mapping a system's security policy requirements to the behavior of the system. As a result of this hierarchy, the phrases

"security policy" and "security model" have taken on a variety of meanings. The five relevant stages are as follows:

1. Higher-level policy objectives
2. External-interface requirements
3. Internal security requirements
4. Rules of operation
5. Top-level specification

A higher-level objective specifies what is to be achieved by proper design and use of a computing system; it constrains the relationship between the system and its environment. The TCSEC control objectives belong to this first level of the requirements hierarchy. An external-interface requirement applies a higher-level objective to a computing system's external interface; it explains what can be expected from the system in support of the objective but does not unduly constrain internal system structure. Internal security requirements constrain relationships among system components and, in the case of a TCB-oriented design, among controlled entities. Rules of operation explain how internal requirements are enforced by specifying access checks and related behaviors that guarantee satisfaction of the internal requirements. A top-level specification is a completely functional description of the TCB interface. It also specifies behavior of system components or controlled entities.

In various contexts, the phrase security policy may refer to any or all of the first four stages of elaboration. In many contexts, the term security policy refers to organizational security policy.

[cf NCSC85, Glossary] From a modeling perspective, organizational policies are the source of higher-level policy objectives. In much of the literature on security modeling, a system security policy is part of the requirements stages of development and provides the definition of security to be modeled. Additional thoughts on the importance of distinguishing among policy objectives, organizational policies, and system policies may be found in Sterne's article "On the Buzzword "Security Policy." [STER91] The terms "AIS security policy" and "automated security policy" are synonyms for "system security policy."

The security policy model is a model of the security policy enforced by the TCB. It is simultaneously a model of the policy and of the system for which the model is given. The portions of the system which are constrained by the model belong to the TCB by definition. The model's definition of security may contain external-interface requirements, but more traditionally consists of internal requirements. Its rules of operation may, in some cases, amount to a top-level specification. There is no firm boundary between rules of operation and top-level specifications: both explain how internal requirements are enforced by the TCB. However, more detail is required

in a TLS, including accurate descriptions of the error messages, exceptions, and effects of the TCB interface. Security requirements occur implicitly in rules of operation and top-level specifications. In this form they are often referred to as security mechanisms. Internal requirements are sometimes classified as "policy" or "mechanism" according to whether or not they are derived from a specific higher-level policy objective.

Conceptually, the security modeling process takes place in two main phases. Requirements modeling takes place after a system security policy is fairly well understood. This is accomplished by constraining the system's design and use based on the higher-level objectives. Rules of operation may be deferred until after the definition of security is established and the basic architecture of the system has been identified.

The following paragraphs contain general suggestions for the construction of security models. Suggestions pertaining to specific kinds of security requirements and specific kinds of systems are given in Sections 3 and 4, respectively.

These suggestions are broken down into six steps that could be carried out with respect to an entire system and security policy or, more simply, for a given component and higher-level policy objective. The first step is to identify externally imposed security policy requirements on how the system (or component) must interact with its environment. The second is to identify the internal entities of the system and to derive the requirements portion of the model in such a way as to extend the external-interface requirements. The third is to give rules of operation in order to show how the modeled security requirements can be enforced. After completion of step three enough information generally is available to determine reliably whether the model is inherently new or can be formalized by known techniques. At classes B2 and above, this is the fourth step. The first three steps taken together result in the identification of several submodels. The fifth step is to demonstrate consistency among these various submodels, so that they fit together in a reasonable way to form a completed security policy model. Finally, the sixth step is to demonstrate relevance of the model to the actual system design. Typically, the majority of the security modeling effort is associated with these last two steps and associated revisions to the model. The total effort needed to produce a security policy model varies considerably, but is often between six staff-months and two staff-years.

2.3.1 STEP 1: IDENTIFY REQUIREMENTS ON THE EXTERNAL INTERFACE

The first step is to identify major security requirements and distinguish them from other kinds of issues. These identified requirements should adequately support known higher-level policy objectives for use of the system. An emphasis on external-interface requirements helps prevent an unrecognized mixing of security and design issues. Such mixing could interfere with the understanding of security and could impose unnecessary constraints on the system design.

External-interface requirements for a computer system can be described in several ways. An elegant, but possibly difficult, approach is to limit the discussion purely to data crossing the system

interface; this is the "black-box" approach. The best known example of the black-box approach is noninterference (see Section 3.2.1). Alternatively, one can describe the system in terms of its interactions with other entities in its environment, such as other computing systems, users, or processes. Finally, one can give a hypothetical description of internal structure that ensures the desired external-interface behavior.

In general, the system's interaction with its environment is constrained by the sensitivity of the information handled and by the authorizations of the individuals and systems accessing the system. Identified user roles, associated privileges, and the extent to which certain roles are security-critical also limit the system's interface to the environment. These constraints determine security attributes that are associated with the system's inputs and outputs. Security attributes may include classification, integrity, origin, ownership, source of authorization, and intended use, among others. The use of such attributes in the construction of security models is discussed in Section 3.

In the case of mandatory access control policies, information must be accurately labeled and handled only by authorized users. This requirement places restrictions on how information is input to the system and, implicitly, on how it will be processed. Authorized handling of information is modeled in terms of constraints on what information may flow from one user to another: information input to the system as classified information should be output only to authorized recipients.

In addition to general regulations, there are often site-dependent and application-dependent constraints that may need to be modeled. In particular, there may be site-dependent constraints on allowed security labels.

Having identified the security requirements on the system interface, it is necessary to decide on the requirements to be covered in the model's definition of security. It must then be decided which of these requirements should be modeled directly or indirectly in terms of internal requirements on system entities and the TCB interface. The set of requirements to include is constrained by minimal TCSEC requirements, the need to adequately support relevant policy objectives, and the need for a simple, understandable model. The inclusion of more requirements may provide more useful information for accreditation once the system is evaluated, but it also increases the difficulty of the vendor's assurance task. The inclusion of more requirements also suggests a more careful structuring of the model in order to show how various aspects of security fit together.

Several factors may influence a decision of whether to directly include external-interface requirements in the security model. The direct inclusion of external-interface requirements can help explain how the model supports higher-level policy objectives. In the case of an application security model, the direct modeling of user-visible operations may be more relevant to end users, a point of view reflected in the SMMS security model. [LAND84] In the case of a network security model, understanding is facilitated by modeling of the network's interaction with hosts in its environment, as will be discussed in Section 4.2.

2.3.2 STEP 2: IDENTIFY INTERNAL REQUIREMENTS

To support the identified external requirements, the system must place constraints on the controlled entities of the system. These internal constraints traditionally form the model's definition of security. In a model whose definition of security contains external-interface

requirements, internal constraints can provide a needed bridge between the definition of security and the rules of operation.

The controlled entities themselves should be identified at a level of granularity that is fine enough to allow needed security-relevant distinctions among entities. At class B2 and above, the controlled entities should include all (active and passive) system resources that are accessible outside of the TCB. For convenience, controlled entities may be grouped into subclasses in any way that facilitates understanding of the system. Such groupings will depend on the system to be modeled. For an operating system, the relevant controlled entities might include buffers, segments, processes, and devices. In most models, it has been convenient to group entities according to whether they play an active or a passive role in order to help show how the TCB implements the reference monitor concept. For networks and other complex systems, identification of controlled entities may need to be preceded by identification of subsystems and their derived security requirements.

Constraints on controlled entities are best stated as general properties and relationships that apply to all (or a broad class of) entities and accesses. Greater generality eliminates unnecessary constraints on the system design, improves one's intuition about the model, and can greatly reduce the overall effort needed to validate correctness of the identified constraints.

The identification of necessary constraints on controlled entities and their interactions is a nontrivial task; familiarity with similar systems and security models can be of great benefit. To define the necessary constraints, it will be necessary to label each entity with appropriate security attributes and to identify possible kinds of interactions, or accesses, between controlled entities. As used here, an access is any interaction that results in the flow of information from one entity to another. [cf DOD88a]

In describing access constraints, it may be useful to know that some kinds of accesses are highly restricted, as when a process accesses a file by executing it. The notion of causality may also be important: a transfer of information from entity e1 to e2 might occur because e1 wrote e2, because e2 read e1, or because a third entity directly copied e1 to e2 without actually reading e1. [cf MCLE90, Sec. 2]

In the particular case of state machine models, constraints often take the form of state invariants, as in the work of Bell and La Padula. Recent efforts suggest that state-transition constraints are an attractive alternative to state invariants for certain kinds of security requirements.

[MCLE88; NCSC90b, Sec. 6.7] The simplest well-known, state-transition constraint is the tranquility principle of Bell and La Padula, which says that the security level of a controlled entity cannot change during a state transition. Tranquility can be artificially rephrased as a state invariant: in any reachable state, the level of an entity coincides with its level in the initial state. Consider, however, the DAC requirement that a subject which gains access to an object must possess the necessary DAC permissions when the access request is made. This is harder to rephrase as a state invariant because DAC permissions can change from one state to the next. Good tutorial examples of state invariants and state-transition constraints may be found in Building a Secure Computer System [GASS87, Sec. 9.5.1, 9.5.2]; more complex examples occur in the TRUSIX work

(NCSC90b, Sec. 2].

The number of internal requirements in a typical model varies considerably depending on the desired level of assurance, the complexity of the policy and system being modeled, and the granularity of the individual requirements. For example, the Trusted UNIX Working Group (TRUSIX) model covers TCSEC access control requirements for a B3 Secure Portable Operating System Interface for Computer Environments (POSIX) system and has eleven state invariants and ten state-transition constraints. [NCSC90b]

The question of which security requirements to cover in the model's internal requirements is answered as much by issues of good engineering practice as by the TCSEC. At a minimum, the model must cover the control of processes outside the TCB. Such processes are potentially a significant security risk because they may be of unknown functionality and origin. According to current practice, those portions of the TCB that do not support user-requested computation do not have to be modeled. For example, the security administrator interface does not have to be modeled.

However, if significant user-requested computation is performed by some portion of the TCB, it should be modeled. A typical example would be a trusted downgrading process available to authorized users (not necessarily the security administrator). A more extreme example would be a multilevel database management system implemented as a TCB subject whose database consisted of a single highly-classified file.

Finally, there is the possibility that some security requirements might be included in the model, but only in the rules of operation. The rules of operation may allow the application of good security practice closer to the implementation level. The rules of operation can then contain security conditions that are not explicitly required by the security policy. Rules of operation can not make up for an inadequate definition of security, however. The task of inferring a modeled security policy from the rules of operation may be excessively difficult, especially if the rules are complex. [cf NCSC90b, Sec. 6.8]

2.3.3 STEP 3: DESIGN RULES OF OPERATION FOR POLICY ENFORCEMENT

How can the modeled security requirements on system entities be enforced? The rules of operation answer this question in broad terms by describing abstract interactions among system entities with particular emphasis on access control and other policy-enforcement mechanisms. In the case of an operating system kernel, the rules of operation would typically describe state transformations and associated access checks needed to uphold the definition of security. In the case of a formal model, this step amounts to the construction of a miniature FTLS. It is a useful preliminary step, even if a full FTLS specification is planned. The rules of operation together with the modeled requirements on controlled entities form the security policy model.

In giving rules of operation, it is important that system behavior be adequately represented. However, actual interactions among controlled entities need not be directly specified when they can be described as a composition of several rules of operation. There is no need for an implementation to directly support an individual rule where several rules have been combined to describe an action. An appropriate level of detail for rules of operation is illustrated by the work of Bell and La Padula. [BELL76] Good tutorial examples may be found in Building a Secure Computer System. [GASS87, § 9.5.1 (Step 3)] More complex examples may be found in the TRUSIX work. [NCSC90b] The rules of operation will usually be more readable if they are not too detailed. At class B3 and above, it is especially important to avoid details that are not security relevant. This is because their inclusion in the model may force their inclusion in the TCB in

order

to achieve a successful model interpretation, thereby violating the TCB minimization requirements. [NCSC85, Sec.3.3.3.1.1]

In the case of a B1 informal model, the rules of operation may reasonably contain information that, for higher evaluation classes, would be found in a DTLS. Thus, a stronger emphasis on policy enforcement than on policy requirements may be useful. This is especially true if the TCB is large and complex, as, for example, when security is retrofitted onto a previously unevaluated system. [BODE88]

A formal model needs to formalize the idea that a particular state transformation is being executed on behalf of a particular subject. Two traditional approaches are to add an extra input to the state transformation that gives the subject id and to add a "current subject" field to the system state. With the former approach, accompanying documentation should explain clearly how the subject-identity parameter is passed, in order to avoid the erroneous impression that the subject is responsible for identifying itself to the TCB. The latter approach suggests that the system being modeled has a single central processing unit. This may be a problem for the model interpretation if the system actually contains multiple CPUs. See the TRUSIX work for further discussion. [NCSC90b, Sec. 6.13]

Finally, in designing rules of operation, it may be convenient to separate access decisions from other kinds of system functionality, including the actual establishment or removal of access.

This separation facilitates exploration of new access control policies. Only the access decision portion is affected by a change in policy because access enforcement depends only on the access decision, not on how it was made. [LAPA90, ABRA90] Isolation of the access decision mechanism occurs in the LOCK system design [SAYD87] and in the security policy architecture of Secure Ware's Compartmented Mode Workstation [NCSC91b, Section 2.2.1.].

2.3.4 STEP 4: DETERMINE WHAT IS ALREADY KNOWN

Usually some, if not all, aspects of the identified security model will have been studied before.

The identification of previously used terminology allows security issues to be presented in a manner that is more easily understood; and making the connection to previously studied issues may provide valuable insight into their successful solution.

For classes B2 and above, the modeling effort must be based on accepted principles of mathematical exposition and reasoning. This is because formal models and mathematical proofs are required. The chosen mathematical formalism must allow for an accurate description of the security model and should provide general mechanisms for its analysis. This is necessary so that specific interactions among the entities of the model can be verified as being appropriate.

In practice, needed mathematical techniques are usually adapted from previous security modeling efforts because security modeling, per se, is generally much easier than the development of new mathematical techniques. Extensive use of past modeling techniques is especially feasible in the case of fairly general and familiar policy requirements. System-dependent modeling is generally required for policy enforcement, but established techniques are often available for demonstrating consistency with the modeled requirements.

If new mathematical techniques are used, their credibility is best established by exposure to

critical review, in order to uncover possible errors in the mathematics and its intended use. This review process is facilitated by the development of comparative results that give useful relationships between new models and old ones.

2.3.5 STEPS: DEMONSTRATE CONSISTENCY AND CORRECTNESS

Since the security provided by a system is largely determined by its security model, it is important that the model capture needed security requirements and that its rules of operation show how to enforce these requirements.

The first crucial step of identifying security requirements on the system interface cannot be directly validated by mathematical techniques. [cf MCLE85] Human review is needed to establish what security requirements need to be addressed and whether these requirements are reflected in later mathematical formalizations. For systems in class B2 and above, real-world interpretations are assigned to key constructs of the formal model in order to provide a common semantic framework for comparing the model and the security policy.

The appropriate technique for validating requirements on controlled entities (Step 2) depends partly on the novelty of the approach. Informal human review is required to assure that the modeled requirements support the original system security policy. Careful comparisons with previous models of the same or related policies may help to show how new modeling techniques relate to previously accepted techniques for handling particular policy concepts. (See, for example, [MCLE87; MCLE90, Sec. 3 & Sec. 4].) An alternate technique is to give an external-interface model and then mathematically prove that the constraints on system entities imply satisfaction of the external-interface model. The external-interface model is easily compared with a security policy or policy objective. The constraints on system entities are those identified in the requirements portion of the security policy model. This technique is illustrated in Section 3.2.

If the rules of operation are given correctly (Step 3), they will comply with the constraints given in the requirements portion of the model. A formal proof of compliance is required for classes B2 and above. In a state-transition model, state invariants are proved by induction: one proves that they hold for the initial state and are preserved by each state transition given in the rules of operation. (See, for example, [CHEH81].) State-transition constraints are straightforwardly proved by showing that each state transition satisfies the constraints.

2.3.6 STEP 6: DEMONSTRATE RELEVANCE

The rules of operation should be a correct abstraction of the implementation. A preliminary model interpretation that is done as part of the modeling process can provide an informal check on whether the rules of operation are sensible and relevant. This model interpretation shows how enforcement mechanisms modeled in the rules of operation are realized by the system. A model interpretation explains what each model entity represents in the system design and shows how each system activity can be understood in terms of the given rules of operation. Thus, for example, a "create file" operation in the system might be explained as involving a restricted use of the rules for "create object," "write object," and/or "set permissions" in the model, depending on the actual arguments to the "create file" command.

An appropriate, somewhat novel, example of a model interpretation is found in the TRUSIX work. [NCSC90b, Sec. 4] The model interpretation is described as an informal mapping from the TRUSIX DTLS to the TRUSIX model. This interpretation first explains how UNIX entities are

represented in the model and the DTLs. For example, messages, semaphores, and shared memory are entities that the DTLs refers to as interprocess communication (IPC) objects but that are treated as "files" in the model. Thirty-six UNIX system interface functions are then described by giving pseudocode that shows how each function can be expressed in terms of the state transformations given by the rules of operation. In other words, the transformations given in the rules of operation form a toy security kernel that could be used to implement UNIX if efficiency were irrelevant.

The SCOMP and Multics model interpretations are examples based on Secure Computer System: Unified Exposition and Multics Interpretation. [BELL76]. In the SCOMP interpretation, a set of security-relevant kernel calls and trusted processes are identified as "SCOMP rules of operation." [HONE85] For each such SCOMP rule, a brief summary of security-relevant functionality is given and then interpreted as the combined effect of restricted use of one or more rules from the above work by Bell. [BELL76] The specific restrictions are enumerated in an appendix, along with a rationale for omitting some kernel calls and trusted functions from the interpretation. The correspondence between the rules in this work [BELL76] and actual SCOMP behavior is rather complex. The Multics interpretation, by way of contrast, is more sketchy, but the actual correspondence appears to be simpler. [MARG85]

2.4 PRESENTATION FOR EVALUATION AND USE

The most important factors in the successful presentation of a security model are the simplicity of the model, its appropriateness, and an understanding of its intended uses.

The presentation of the model must demonstrate that the model correctly describes the system security policy. A clear explanation of the policy from which the model is derived should be available for this demonstration. Sufficiency of the model may be demonstrated by presenting the relationship of the model to the policy and by carefully explaining and justifying the modeling decisions behind this relationship. In addition to modeled requirements, the system may support other security requirements that are not suitable for inclusion in the model, especially if it is a formal model. Unmodeled security requirements can be presented in an appendix so that TCB developers have all of the security requirements in a single document.

An overview of the model interpretation is needed so that readers can understand the relevance of the system's security model to the security policy and to the overall system-development process. All of these topics may be legitimately covered in a philosophy of protection. In fact, it is highly desirable that the philosophy of protection be the first document delivered by the vendor in a system evaluation, so that it can serve as a basis for further understanding of the security model and other system documentation.

In the case of a formal model, an informal explanation or presentation of the model prepared for a general audience is highly desirable. This is partly for reasons mentioned in Section 2.3.5 and

partly because of the general need to acquaint system developers, implementors, and end users with the basic security principles that are to be supported by the system. The informal presentation can reasonably be included in the philosophy of protection.

A well-written explanation of the model's definition of security can be used by potential buyers of a secure system to determine compatibility between the computing system and their organization's security policies and needs. To evaluate the ability of a computing system to support these policies, potential users may wish to construct an informal model of their security policies

and compare it to the definition of security supported by the computing system. This use of the security model and the fact that some aspects of security modeling can only be validated by social review suggest that portions of the security model and the philosophy of protection be made available to a relatively wide audience, for example, treating them as publicly released, nonproprietary documents.

For systems at the B2 level or above, the requirement of mathematical proof necessitates the presentation of a rigorous mathematical formalization as well. For a mathematical audience, standards of precision may preclude a style that will be appropriate for a general audience. In this case, an informal presentation of the model is also needed in order to reach the general audience.

At the A1 level of assurance, the formal model is directly involved in the formal verification of the FTLS. As a result, it may be appropriate to present the model in the formal specification language of an endorsed verification system (see Appendix B). Authors and readers must be fully aware of the nuances of the descriptive notation of the verification system used in order to avoid errors due to discrepancies between author intent and reader expectation. Furthermore, if a formal verification system is used to check proofs, it is necessary to achieve a correct translation of what is to be proved into the language of the verification system. Further discussion of these points may be found in [FARM86; NCSC90b, Sec. 6.3].

3. SECURITY MODELING TECHNIQUES

Having introduced the major topics involved in security modeling, we now consider detailed modeling issues with emphasis on mathematical structure as well as empirical justification. In this section, a review of basic concepts is followed by discussions on the modeling of various kinds of policies and objectives: nondisclosure, need-to-know, integrity, and special-purpose policies of particular TCB subjects.

In most cases, modeling techniques are introduced, with details handled via references to the available literature. Inclusion of these references does not imply NCSC endorsement of referenced products incorporating the techniques discussed.

3.1 BASIC CONCEPTS

As discussed in Section 2, the identification of controlled entities plays a crucial role in the development of a security policy model. At B2 and above, the controlled entities must include all system resources. If the policy is multifaceted, with separate subpolicies for mandatory and discretionary access controls, it may be acceptable to decompose the system into different sets of controlled entities for different subpolicies. A secure computing system may decompose into data structures, processes, information about users, I/O devices, and security attributes for controlled entities. In general, the number of different kinds of entities depends on what security-relevant distinctions are to be made. The following paragraphs discuss how to perform such a decomposition for typical kinds of entities, with the goal of modeling security requirements in such a way as to allow an accurate, useful model interpretation.

An explicitly controlled entry is one that has explicitly associated security attributes. In the TRUSIX model, for example, the explicitly controlled entities are referred to as "elements". They consist of subjects and three kinds of objects; files, directories, and entries (i.e., links). [NCSC90b, Sec. 6.9-6.11] In addition to explicitly controlled entities, a system will have implicitly controlled entities. Such an entity might be contained in an explicitly controlled entity or might be a composite entity composed of explicitly controlled entities having potentially different security attributes.

The discussion of security attributes in this section emphasizes security levels used for mandatory access control because the TCSEC labeling requirements apply primarily to mandatory access control and auditing. However, much of what is said about MAC labels applies to other kinds of security attributes including, for example, discretionary access control lists.

It is necessary to model creation and destruction of most kinds of controlled entities. A simpler model results if the same approach is used in all cases. In a formal model, one can model the creation of new controlled entities by modeling changes in the set of all controlled entities. A fixed set of possible controlled entities, together with an indication of which entities are available for use in a particular state (e.g., which subjects are active, which objects or object-ids are allocated) can also be used. [cf NCSC90b]

3.1.1 DATA STRUCTURES AND STORAGE OBJECTS

In this guideline, a data structure is a repository for data with an internal state, or value, that can be written (i.e., changed) and/or read (i.e., observed) by processes (and, possibly, devices) using well defined operations available to them. A data structure is distinct from its value at a particular time, which is, in turn, distinct from the information conveyed by that value. The information content depends not only on the value but also on how it is interpreted. Similarly, a data structure is distinct from a name for that data structure.

A data structure that is explicitly associated with exactly one security level by the security policy model is a storage object. There are no a priori restrictions on the level of abstraction at which data structures and storage objects are modeled. In particular, objects may be abstract data structures that are accessed using high-level operations provided by an encapsulation mechanism. In this case, a user would have access to the high-level operations but could not access the encapsulated data directly via more concrete operations that may have been used to define the high-level operations. This lack of direct access would have to be enforced by the TCB.

Ordinarily, the storage objects in a security model are disjoint. This means that, in principle, changes to one object do not force changes to other objects. If a state-machine model is used, disjoint storage objects can be modeled as separate components of the underlying machine state. If two objects at different security levels were not disjoint, then access to data in their intersection would be allowed or denied according to which label is used. This ambiguity can be resolved by explicitly associating a level with their intersection. In this case, the associated level

allows the intersection to also be regarded as a separate storage object, thereby restoring disjointness. Thus, disjointness in storage objects simplifies one's understanding of how security labels are used to control access. It has also been argued that disjointness simplifies covert channel analysis via shared resource matrices.

In some systems, collections of objects are combined to form multilevel data structures that are assigned their own security levels. Multilevel data structures called "containers" are used in the Secure Military Message System (SMMS) to address aggregation problems. [LAND84] The level of a container must dominate the levels of any objects or containers inside it.

When an object or other controlled entity is created, it must be assigned security attributes and initialized in such a way as to satisfy the object reuse requirement. The modeling of object reuse policies (e.g., objects are erased when allocated) can be useful, but object reuse is not found in traditional access control models because the object reuse requirement deals with the content of objects and TCB data structures. The initialization of security attributes, however, plays an essential role in access control and can be modeled by distinguishing between "active" and "inactive" entities, as in [NCSC90b]. The security attributes of a newly created object may be taken from, or assigned by, the subject that created it. In many systems, creating an object is essentially a special case of writing to it: its value changes from undefined to null. If this change is visible to non-TCB subjects and the new object is created by a non-TCB subject, then the security level of the new object must dominate that of the creating subject. DAC attributes, by way of contrast, are usually given by system- or user-supplied defaults.

3.1.2 PROCESSES AND SUBJECTS

A process may create, destroy, and interact with storage objects and other processes, and it may interact with I/O devices. It has an associated run-time environment and is distinct from the program or code which defines it. For instance, the program would ordinarily be modeled as information contained in a storage object. An explicitly controlled process is a subject. It normally has a variety of associated security attributes including a security level, hardware security attributes such as its process domain, the user and user group on whose behalf it is executing, indications of whether it belongs to the TCB, and indications of whether it is exempt from certain access control checks.

The issues regarding disjointness of storage objects mentioned in Section 3.1.1 apply to subjects as well. If two subjects share memory, it is advisable to model the shared memory as an object separate from either subject in order to achieve the disjointness needed for information-flow analysis. Local memory need not be separately modeled, but must be properly taken into account when modeling the subject. Registers (including the instruction pointer) are technically shared memory but can often be treated abstractly as unshared local memory. This is especially true if registers are saved and restored during process swaps and the information they contain is not shared between processes.

Security levels for processes are ordinarily similar to security levels for storage objects. The

TCSEC requires that a subject have a nondisclosure level that is dominated by the clearance and authorization of its user. A TCB subject may reasonably be assigned separate levels for reading and writing in order to allow partial exemption from access control (see Section 3.4). Other TCB-related entities may also need separate levels for reading and writing in some systems. One such example is /dev/null in UNIX. [GLIG86] This object is system high in the sense that any process can write to it; but it is system low in the sense that any process can read from it because its value is always null.

In general, the security-relevant attributes of a subject are part of its run-time environment. Some attributes are relatively static and are inherited from the subject's executable code or are stored with its code. In UNIX, the property of being a set-user-id program is a statically determined attribute, as is the effective user-id. Other attributes are dynamically assigned and are inherited from a subject's parent or are actively assigned by the parent (assuming it has a parent process). In the latter case, the parent must be relied upon to assign the child's security attributes appropriately. As a result the parent usually belongs to the TCB in this latter case. A secure terminal server, for example, might create a subject and assign it security attributes based on the determined identity of a user logging in.

There are potential difficulties in associating user-ids with TCB subjects. A terminal server would appear to operate on behalf of the user currently logged in or on behalf of the Identification and Authentication mechanism if no one was logged in. A print spooler acts on behalf of the users in its print queue. Such TCB subjects are, in reality, associated with multiple user-ids, a fact which is relevant to the design of the audit mechanism. A TCB subject that does not act on behalf of a given user can be handled either as having a dynamically modifiable user-id or as acting on behalf of the system (equivalently, as acting on behalf of a system-defined pseudo-user).

When a subject creates a child subject by executing a program, the resulting child might belong to the TCB and be exempt from certain access control checks, even if the original subject was not in the TCB. This is possible if exemptions are associated with the program object itself.

With the advent of multitasking languages such as Ada, there is a question of when two threads of control belong to the same subject. Pragmatically, to be the same subject, they should have the same security attributes. To qualify as separate subjects, their separation should be enforceable by the TCB. Finally, nothing explicitly prohibits a multithreaded process from consisting of several subjects operating at different security levels. Of course, intraprocess communication will be somewhat limited for such a process unless it contains TCB subjects that are exempt from some of the MAC constraints.

3.1.3 USERS AND USER ROLES

User-related topics often turn up in security models in spite of the fact that users are not controlled entities and thus do not need to be directly modeled. The users of a system may perform specific user roles by executing associated role-support programs. In general, a user may engage

in a combination of several roles. As a matter of policy, a given user role may require system-mediated authorization and may provide specific system resources needed for performance of the role. Although the following paragraphs discuss only individual user roles, most of the basic concepts extend to roles for other kinds of clients in the environment of a computing system including user groups and, in the case of a network, hosts or subjects running on hosts.

The TCSEC requires support for certain trusted user roles. For systems at B2 and above, these roles include the security administrator role and the system operator role. The administrator role governs security attributes of users, moderates discretionary and mandatory access control, and interprets audit data. The operator role ensures provision of service and performs accounting activities. [NCSC90a] These roles may be subdivided so that they can be shared by multiple users. In addition, they do not preclude the addition of other trusted roles. In general, trusted user roles are characterized by the fact that they involve handling security-critical information in a way that violates access restrictions placed on non-TCB subjects. As a result, processes that support trusted roles can be misused and must have restricted access. By definition, such trusted role processes have security properties that are unacceptable without special constraints on the behavior of their users. These observations suggest (but do not mandate) the modeling of support for user roles, especially trusted user roles. Trusted role processes are usually treated as TCB subjects. Additional information on modeling them is given in Section 3.4.

Instructive examples of role definitions may be found in Secure Xenix [GLIG86] and the SMMS security model [LAND84]. Secure Xenix restructured the Guru role into four separate roles. The SMMS security model included a system security officer role, a downgrader role and a releaser role. In both of these systems, the roles support separation of duty in that each role has privileges not available to the other roles. Separation of duty has been emphasized by Clark and Wilson, who have given a general scheme for constraining user roles by using triples of the form (user, program, object-list) in order to control how and whether each user may access a given collection of objects. [CLAR87] A similar role-enforcement mechanism is supported by type enforcement. [BOEB85] The type-enforcement mechanism itself assigns "types" to both subjects and objects; subject types are referred to as "domains." The accesses of each subject are constrained by a "type table" based on subject domain and object type. Each user has a set of domains associated with subjects running on his behalf. These domains are used to define user roles. [cf THOM90]

3.1.4 I/O DEVICES

Although users are external to the computing system and need not be directly modeled, it is still useful to model their allowed interactions with the system in order to document security-relevant constraints that are enforced by the system. User interactions may be modeled in terms of constraints on I/O devices if there is a convention for discussing the current user of a device. Devices which are accessible to subjects outside the TCB should be modeled either implicitly or explicitly as part of the interface between the non-TCB subjects and the TCB. An additional reason for interest in I/O devices is that I/O typically accounts for a large portion of the TCB. The following paragraphs present general information on the modeling of devices with emphasis on device security requirements and then discuss when they should be modeled as objects, as subjects, or as their own kind of entity.

Normally, security models discuss abstract encapsulated devices rather than actual physical

devices. An encapsulated device is an abstract representation of a device, its associated device driver, and possibly other associated entities such as attached hardware and dedicated device buffers. Thus, for example, one might discuss a line printer connected to the system but not the actual RS-232 device used to achieve the connection or its associated device driver.

By definition, an input device injects information into the system in a way that the system cannot entirely control. The next state of the system may depend on the actual input as well as on the current state and the particular state transformation being executed. This fact can be accommodated in several ways in an FTLS or a model that addresses object content. One can treat the actual input as a parameter to the transformation. If there are only a few different input values, one can associate different transformations with different inputs. Finally, one can treat the transformation as being nondeterministic, meaning that several different states can result from executing the transformation in a given state.

Abstract encapsulated devices are often passive entities, in contrast to their underlying hardware. Security requirements for devices, however, differ significantly from those for either storage objects or controlled processes:

- External policy on use of the system requires that devices pass information only to authorized users.
- Devices may transport either unlabeled data or labeled data and are classified as single level or multilevel devices accordingly.
- At B2 and above, the TCSEC requires that every device have a minimum and maximum device level that represents constraints imposed by the physical environment in which the device is located.

Authorized use of a device may be enforced by requiring that any piece of information output by the device have a security level that is dominated by the clearance and authorization of the recipient. A combination of procedural and automated methods are needed to correctly associate the security level of the information, the user who will receive that information, and the security level of that user. Typically, the device itself will also have a security level. The user who receives information from a device may well be different than the user who sent that information to the device. If a device with local memory (an intelligent terminal, for example) is allocated to different users or different security levels at different times, it will typically need to be reset between users in order to meet requirements on authorized transmission of information.

Both single-level and multilevel devices may handle multiple levels of information. A single-level device can only handle a single level at a time but may have some convention for changing levels. A multilevel device, by way of contrast, can handle different levels of data without altering its security designation, but still might be used in such a way as to carry data at only one fixed security level.

The TCSEC requirement for device ranges does not explain how they are to be used. The most common option is to require that the level of any object transmitted by the device be within the range. Another option is to require that the security clearance of any user be within the range. Separate decisions regarding device minimum and device maximum are also possible. In the Trusted Xenix system, a user with a secret clearance can have an unclassified session on a terminal with a secret device minimum. The intended interpretation of the device minimum is that the device is in a restricted area that should contain only users whose clearance is at least

the device minimum. If the device minimum is secret, then a login attempt by an uncleared user is treated as a violation of physical security. [GLIG86]

The question of whether devices should be modeled in the same way as other kinds of controlled entities depends on the complexity of the devices involved, observed similarities between devices and other modeled entities, and goals relating to the simplicity and accuracy of the model. The TRUSIX group decided to model devices in the same way as file objects, a decision that depended heavily on both the nature of UNIX and the specific goals of the TRUSIX effort. [NCSC90b, § 6.9] The SMMS model treats devices as containers in order to capture the fact that the device maximum must dominate the level of all data handled by the device. [LAND84] Yet another alternative is to model devices by modeling their device drivers as TCB subjects. In general, more sophisticated I/O devices need greater modeling support, with the limiting case being intelligent workstations modeled as autonomous computing systems.

3.1.5 SECURITY ATTRIBUTES

A security attribute is any piece of information that may be associated with a controlled entity or user for the purpose of implementing a security policy. The attributes of controlled entities may be implicit; they need not be directly implemented in data structures. Labels on a multilevel tape, for example, can be stored separately from the objects they label, provided there is an assured method of determining the level of each object on the tape. [cf NCSC88b, C1-C1-05-84]

Primary attributes of security policies that are usefully reflected in the security model include locus of policy enforcement, strength and purpose of the policy, granularity of user designations, and locus of administrative authority. These policy aspects lead to a taxonomy of policies and security attributes.

There are several types of security attributes of any given system: informational attributes, access control attributes, nondisclosure attributes, and integrity attributes. Informational attributes are maintained for use outside the given computing system, whereas access control attributes limit access to system resources and the information they contain. Purely informational attributes are somewhat uncommon; an informative example is given in Section 4.4. Access control attributes may be classified according to what they control. A loose access control attribute controls access to the entity it is associated with, whereas a tight access control attribute also controls access to information contained in that entity. Thus, access restrictions determined by tight attributes must propagate from one object to another when (or before) information is transferred, because control over the information must still be maintained after it leaves the original entity. Nondisclosure attributes are used to prevent unauthorized release of information, whereas integrity attributes are used to prevent unauthorized modification or destruction of information.

Attributes can also be classified by the granularity and authority of their control. The user granularity of an attribute may be "coarse," controlling access on the basis of broadly defined classes of users, or it can be "per-user," controlling access by individual users and processes acting on their behalf. Finally, centralized authority implies that policy for use of attributes is predefined

and takes place under the control of a system security administrator, whereas distributed authority implies that attributes are set by individual users for entities under their control. This classification of security attributes is based partly on the work of Abrams. [ABRA90]

When applied to typical security policies for B2 systems, these distinctions among security attributes might take the form given in Figure 3.1. MAC policies must enforce nondisclosure constraints on information and may enforce integrity constraints as well. They must regulate access to information on the basis of user clearance and labeled sensitivity of data. The involvement of user clearances typically comes at the expense of per-user granularity because several users are likely to have the same clearance. Authority to assign security attributes is usually somewhat centralized. Users can create entities at various levels, but ability to change the levels of existing entities is usually restricted to authorized users.

DAC policies must enforce access constraints on both reading and writing. They must provide per-user granularity as criteria for access decisions, although they often include a group mechanism for coarse grained access decisions as well. DAC policies are normally used to enforce both nondisclosure and integrity, but constraints on access to named objects do not necessarily imply corresponding constraints on access to information in those objects. Authority to change discretionary attributes is usually distributed among users on the basis of ownership.

	MAC	DAC
Binding Strength:	Tight	Loose
Purpose:	Nondisclosure	Nondisclosure & Integrity
User Granularity:	Coarse	Per-user
Authority	Centralized	Distributed

Figure 3.1. Typical Classification of Access Control Policies

3.1.6 PARTIALLY ORDERED SECURITY ATTRIBUTES

A security attribute belonging to a partially ordered set may be referred to as a level. The associated partial ordering may be referred to as the dominance relation; in symbols, L_1 is dominated by L_2 if and only if $L_1 \leq L_2$. The use of partially ordered levels is usually associated with tight access control policies and constraints on information flow. Constraints on information flow can arise for several different reasons. As a result a multifaceted policy might have a different partial ordering for each reason. Often, the combined effect of constraints associated with several partial orderings can be expressed in terms of a single composite ordering. In this case, facts about Cartesian products of partially ordered sets given in Appendix A may be used to simplify the formal model and, possibly, the system's implementation as well. The following paragraphs discuss the connection between levels and constraints on information flow, the use of these constraints for supporting nondisclosure and integrity objectives, and the tailoring of level-based policies to particular applications.

A dominance relation is often viewed as an abstraction of allowed information flows:

information can flow from entity E1 to entity E2 only if the level of E1 is dominated by the level of E2. This rather general view, which is an analogue of the original *-property of Bell and La Padula, allows the illustration of some basic issues in the use of levels, but it is overly simple in some respects. It does not address whether information flows properly or whether the flow is direct or indirect. Moreover, this view does not contain any explicit assumption about why information may be prevented from flowing from one entity to another.

Partially ordered levels and related constraints on information flow have been suggested for use in enforcing both nondisclosure and integrity objectives. The motivation for these suggestions may be understood from the following considerations. Suppose that access controls could enforce the above information-flow constraints perfectly. Suppose that the two objects contain the same information, but one is labeled at a higher level than the other. In this situation, information is less likely to be disclosed from the higher-level object because fewer subjects have a sufficiently high level to receive this information. Conversely, if lower-level subjects can write higher-level objects, then inappropriate modification of the lower-level object is less likely because fewer subjects have a sufficiently low level to write to it. This dual relationship may cause the goals of nondisclosure and integrity to conflict, especially if the ordering on levels is strongly hierarchical. As explained in Section 3.5, this duality between nondisclosure and integrity has some significant limitations. A given set of security levels designed for enforcing nondisclosure may be inappropriate for enforcing integrity, and not all integrity policies use partially ordered security attributes.

When a computing system is accredited for use at a particular site, it is assigned a host accreditation range - a set of levels at which the host may store, process, and transmit data. A host range can prevent the aggregation of certain classes of data. If several objects have different levels and no level in the host range dominates all of them, then there is no legitimate way of concatenating these objects. This use of host ranges is discussed further in Section 4.4.

A desirable constraint holds between a host range and device ranges: if the host range contains a level that does not belong to any device range, a non-TCB subject running at that level can not communicate bidirectionally without using covert channels. By way of contrast, a device range may reasonably contain levels not in the host range. Suppose, for example, that A and B are incomparable levels dominated by a level C that does not belong to the host range. A device might reasonably use C as a maximum device level in order to allow use of the device at either level A or level B. But, in this case, the system would need to check that each object input from the device actually belonged to the host range.

3.1.7 NONDISCLOSURE LEVELS

The following paragraphs discuss the structure of nondisclosure levels as it relates to their abstract mathematical properties, to TCSEC requirements, and to their intended use. Analyses given in the following paragraphs suggest that the structure of nondisclosure levels that are used to enforce nondisclosure policies is often not needed for modeling purposes. If their structure

plays

no significant role in a given model, their inclusion is unnecessary and may limit the applicability of the model.

The TCSEC requires that nondisclosure levels contain a classification component and a category component. The hierarchical "classification" component is chosen from a linearly ordered set and the nonhierarchical "category" component must belong to a set of the form $*(C)$, the set of all subsets of C , for some set C of categories. [cf NCSC85, Sec. 9.0, Sec. 3.1.1.4]

In some applications, thousands of categories may be needed. In others, only the four clearance levels "unclassified", "confidential", "secret", and "top secret" are needed, as a result of Executive Order

12356. [REAG82] These facts illustrate the importance of configuration-dependent host accreditation ranges to remove unused security levels.

As explained in Appendix A, any partially ordered set may be fully embedded in one based on categories. As a result, TCSEC constraints on nondisclosure levels do not restrict the class of

partially ordered sets that may be used for such levels (although these constraints do affect the use of human-readable names). The decomposition of levels into classification and category components can be configuration-dependent and may be given along with the host accreditation range. This fact is of some interest for computing systems with both commercial and military applications. [cf BELL90] In this case, the model should be general enough to embrace all intended configurations.

Even if labels are explicitly assumed to contain classification and category components, nothing in the TCSEC prevents the addition of vendor-supplied components because such additions do not invalidate the mandated access checks. Thus, for example, each label could contain a release date after which the contained information will be valueless and, therefore, unclassified. If release dates are chronologically ordered, later dates would represent a higher nondisclosure level. The initial specification of release dates, like that of other nondisclosure attributes, needs to be handled by trusted path software. The regrading from classified - outdated to unclassified would be carried out by trusted software in conformance with accepted downgrading requirements.

A nondisclosure level is, by definition, commensurate with the level of harm that could result from unauthorized disclosure, and it should also be commensurate with the level of assurance against unauthorized disclosure that is to be found in the system's TCB and surrounding physical environment. Various rules of thumb have been worked out to correlate risk with levels of nondisclosure and assurance. [DOD88a, NCSC85a] While these relationships are not likely to show up in the security model itself, they may affect the kinds of security attributes that are included in the nondisclosure level.

3.1.8 UNLABELED ENTITIES AND THE TRUSTED COMPUTING BASE

In addition to controlled entities, there are system resources that either are not user accessible or are part of the mechanism by which user-accessible resources are controlled. These latter resources are the software, hardware, and internal data structures which make up the TCB. At higher levels of assurance, significant effort goes into minimizing the size and complexity of the TCB in order to reduce the overall effort needed to validate its correct operation.

The TCB typically consists of the implemented access control mechanism and other entities

that must be protected in order to maintain the overall security of the system. A TCB process which is not part of the access control mechanism may be assigned security attributes and controlled as a subject in order to help enforce the principle of least privilege within the TCB. Conversely, certain subjects may need to be included within the TCB because they assign security levels to input, support trusted user roles, transform data in such a way as to legitimately alter the data security level, or perform some other security-critical function. A data structure may also be security-critical, as when it contains user-authentication data, a portion of the audit trail, audit-control data, or security attributes of controlled entities. It is not always necessary to assign security attributes to TCB processes and security-critical data structures, but this is often done to enforce the principle of least privilege within the TCB and to regulate access by non-TCB subjects to security critical data structures.

If a piece of data can be accessed as a direct effect of a system call (i.e. access directly specified in a parameter) then it must be accounted for in the interpretation of controlled entities in such away as to satisfy MAC requirements. But some data structures may not be directly accessible. Possible examples include security labels, the current access matrix, internal object names that are not accessible to users of the system, and transient information related to access control, opening of files, and so forth. A data structure which is not directly accessible does not have to be labeled. A decision to supply a label may complicate the modeling process, whereas a decision not to supply a label may increase the difficulty of the covert channel analysis.

While there is no explicit requirement to model the TCB, the model must capture security requirements imposed by the TCSEC, including reference monitor requirements relating to non-TCB subjects and the entities they manipulate. [cf NCSC87, Appendices B.3.4, B.7.1] A possible approach to modeling these reference monitor requirements is disc

Share this article



Receive all the latest articles by email!

Receive Real-Time & Monthly WindowSecurity.com article updates in your mailbox. Enter your email below!
Click for [Real-Time sample](#) & [Monthly sample](#)

Become a WindowSecurity.com member!

Discuss your security issues with thousands of other network security experts. [Click here](#) to join!

[About Us](#) : [Email us](#) : [Product Submission Form](#) : [Advertising Information](#)

WindowsSecurity.com is in no way affiliated with Microsoft Corp. *Links are sponsored by advertisers.

Copyright © 2008 [TechGenix Ltd.](#) All rights reserved. Please read our [Privacy Policy](#) and [Terms & Conditions](#).