UK IT SECURITY EVALUATION AND CERTIFICATION SCHEME

UK Scheme Publication No 4

DEVELOPERS' GUIDE

Part III

ADVICE TO DEVELOPERS

Issue 1.0

July 1996

Issued by:-

UK IT Security Evaluation & Certification Scheme

Certification Body

FOREWORD

The UK IT Security Evaluation and Certification Scheme has been established to evaluate and certify the trustworthiness of security features in Information Technology (IT) products and systems.

The Developers' Guide provides guidance to developers and sponsors on how to ensure that the development of secure products and systems meet the evaluation and certification requirements of the IT Security Evaluation Criteria (ITSEC).

This document (Part III of the Developers' Guide) provides advice for sponsors and developers on specific topics of the ITSEC for secure products and systems.

P. M. Seeviour
Senior Executive
UK IT Security Evaluation and Certification Scheme

Correspondence in connection with this Guide, including requests for additional copies, should be addressed to:

Certification Body Secretariat
UK IT Security Evaluation & Certification Scheme
Certification Body
PO Box 152
Cheltenham
Gloucestershire
GL52 5UF
United Kingdom

Telephone: +44 1242 238739

Facsimile: +44 1242 235233

E-mail: CBSec@itsec.gov.uk

UK IT Security Evaluation & Certification Scheme
Developers' Guide - Part III: Advice to Developers

AMENDMENT RECORD

Amendments to this document will be published as and when required. The amendment record shall be maintained so that it indicates all changes made to the latest issue of the document.

Amendment Instruction Number	Pages Affected	Incorporated by	Date
		Name Signature	

CONTENTS

FIGURES

REFERENCES

A ITSEC - Information Technology Security Evaluation Criteria, Provisional Harmonised Criteria, Version 1.2, Commission of the European Communities, 28 June 1991

B ITSEM - IT Security Evaluation Manual, Version 1.0, Commission of the European Communities, 10 September 1993

C MEMO 5 - CESG Electronic Information Systems Security Memorandum No. 5, System Security Policies, Issue 1.0, May 1992

D TCSEC - Trusted Computer Systems Evaluation Criteria, DoD 5200.28-STD, Department of Defense, United States of America, December 1985

E A Note on the Yourdon Structured Model, ACM SIGSOFT Software Engineering Notes Vol. 15 No. 2, p. 27, A. J. Bowles, Yourdon Inc., April 1990

F An Introduction to SADT Structured Analysis and Design Technique, Report 9022-78R, SofTech Inc., 460 Totten Pond Road, Waltham, MA 02154, USA, November 1976

G The SSADM Manual, ISBN 085-012-527-X, National Computing Centre Ltd., Manchester, UK, 1989

H System Development, M. A. Jackson, Prentice Hall International, 1983

I Principles of Program Design, M. A. Jackson, Academic Press, New York, USA, 1983

ABBREVIATIONS

ACL	Access Control List
API	Application Programming Interface
BS	British Standard
CASE	Computer Aided Software Engineering
CESG	Communications-Electronics Security Group
CLEF	Commercial Licensed Evaluation Facility
CMS	Certificate Maintenance Scheme
CMW	Compartmented Mode Workstation
COTS	Commercial-Off-The-Shelf
DAC	Discretionary Access Control
DTLS	Descriptive Top Level Specification
EN	European Norm
EOR	Evaluation Observation Report
ETR	Evaluation Technical Report
HMG	Her Majesty's Government
ISO	International Standards Organisation
IT	Information Technology
ITSEC	Information Technology Security Evaluation Criteria
	Information Technology Security Evaluation and Certification (see Scheme)
ITSEM	Information Technology Security Evaluation Manual
MMI	Man Machine Interface
PDL	Program Description Language
RDBMS	Relational Database Management System
Scheme	UK IT Security Evaluation and Certification Scheme
SEISP	System Electronic Information Security Policy
SEF	Security Enforcing Function
SFN	Security Fault Notification
SoM	Strength of Mechanisms
SSADM	Structured Systems Analysis and Design Method
SSP	System Security Policy
TCB	Trusted Computing Base
TCSEC	Trusted Computer System Evaluation Criteria (known as the *Orange Book*)
TOE	Target of Evaluation
UK	United Kingdom
UKAS	UK Accreditation Service
UKSP	UK Scheme Publication
US	United States of America

Chapter 1 Introduction

How to use this Guide

1.1 The Developers' Guide has been produced to assist developers intending to submit their products or systems for evaluation under the UK IT Security Evaluation and Certification Scheme ('the Scheme').

1.2 The Guide is divided into three parts:

a) Part I - Roles of Developers in ITSEC

b) Part II - Reference for Developers

c) Part III - Advice to Developers.

1.3 Part I provides an introduction to the ITSEC for developers. It emphasises the roles and responsibilities of developers and their interactions with other organisations within the Scheme. Readers of this Guide who are familiar with the evaluation process under the Scheme and the basic requirements placed upon developers may wish to concentrate on Parts II and III.

1.4 Part II provides a detailed guide to the ITSEC criteria which are relevant to developers.

1.5 Part III (i.e. this document) gives advice to developers on how to tackle development issues which are specific to the ITSEC criteria.

1.6 In addition, a Developers' Guide Roadmap provides an explanation of the Guide's documents to readers, with suggested reading plans.

Objective of Part III

1.7 This part of the Developers' Guide provides advice on ITSEC topics to enable developers and sponsors of IT products and systems to prepare deliverables which will satisfy the criteria.

1.8 This guide does not replace or supersede the ITSEC requirements as specified by the ITSEC [Reference 0] and the ITSEM [Reference 0].

Scope

1.9 The topics in this part of the Guide cover a range of issues in more depth than Part II.

1.10 Readers of this part of the Guide should be familiar with the material contained in Parts I and II.

Acknowledgements

1.11 The Developers' Guide contains excerpts from the Information Technology Security Evaluation Criteria (ITSEC) [Reference 0] and the IT Security Evaluation Manual (ITSEM) [Reference 0], both published by the Commission of the European Communities.

A Request for Feedback

1.12 Developers and sponsors play a significant part in the performance and success of the Scheme. It is the UK Certification Body's wish to assist developers and sponsors with their understanding of the Scheme to enhance the likelihood of successful evaluations and further increase the number of certificates awarded.

1.13 The Certification Body extends an invitation to all developers and sponsors to contact the Certification Body with feedback on this Guide and the Scheme or to obtain further advice on the Scheme.

1.14 Please contact the Certification Body at the address shown in the Foreword (see page 5).

Chapter 2 Writing a Security Target

Purpose

2.1 This chapter gives guidance on the preparation and layout of a security target, suitable as input to an evaluation to the ITSEC evaluation criteria. It is aimed at sponsors who intend to submit their products or systems for evaluation under the UK IT Security Evaluation Scheme. The main difference between an evaluation of a product and a system is that the product security target includes a product rationale, whilst a system security target will reference its System Security Policy (SSP). These issues are discussed below.

2.2 Each explanatory section of this chapter will reference two examples, contained in Annexes 0 and 0. Both examples are based on products and this chapter will concentrate on the requirements for product security targets, although an explanation of the System Security Policy is included. Annex 0 contains an example of a security target for an E1 evaluation using explicit definitions of the security enforcing functions. Annex 0 contains an example of a security target for an E3 evaluation, with the functionality being defined using the predefined functionality class F-C2.

Definition of the Security Target

2.3 The objectives of the security target are:

a) to provide a specification of the security functionality of a target of evaluation (TOE)

b) to relate a TOE to the environment in which it is intended to operate

c) to provide the basis of the evaluation.

2.4 The intended audience for a security target may therefore include:

a) the developer of the TOE - the security target defines the security requirements of the TOE

b) the evaluators - the security target provides the baseline against which the TOE is evaluated

c) the vendor or user of a TOE - the security target specifies the security objectives of the TOE to those responsible for managing, purchasing, installing, configuring and operating the TOE

d) accreditors - the security target is used in conjunction with the System Security Policy as the basis on which accreditation of the system is granted.

2.5 The security target is the basis for the evaluation and is itself subject to evaluation.

2.6 The required content of the security target is determined by whether the TOE is a system or product. A system operates under a known set of operational constraints within a specific environment. For a product the constraints and environment are not known, and must be assumed by the security target. The required content can be summarised as follows:

a) either a System Security Policy or a product rationale

b) a specification of the required security enforcing functions

c) a definition of the required security mechanisms (optional)

d) the claimed rating of the minimum strength of mechanisms

e) the target evaluation level.

The System Security Policy (SSP)

2.7 A System Security Policy (SSP) specifies the rules and procedures which need to be applied to ensure an adequate level of security for a system. The SSP defines the system in terms of its hardware configuration, its interface with other systems, and the national or organisational protective marking of data the system will handle. It also specifies the type of users and the operational role intended for the system. Having defined the system the SSP identifies its security objectives and the threats posed to them. The SSP also describes the set of security measures which are designed to satisfy system objectives and counter identified threats, together with responsibilities for their implementation.

2.8 At a lower level a System Electronic Information Security Policy (SEISP) provides more detailed information on the technical aspects of a system, forming a comprehensive and self consistent statement of the measures required to enforce security. CESG Electronic Information Systems Security Memorandum No. 5 [Reference 0] suggests that for a system, a properly constructed SSP and SEISP, taken together, should be capable of meeting the requirements for a security target. Memorandum No. 5 also contains detailed guidance on the production of these documents.

2.9 Guidance on producing both an SSP and an SEISP can be found in Memorandum No. 5 [Reference 0]. More specific assistance on how to relate this guidance to the ITSEC requirements for a security target can be obtained from the Certification Body.

The Product Rationale

2.10 In the case of a product, the developer may not know the precise environment in which the product will be used, since it may be incorporated into more than one system or environment, or even into another product. Therefore a statement, or product rationale, is provided which gives the prospective purchaser the information needed to decide whether the product meets the security objectives for their system or product.

2.11 The product rationale should give a description of the type of environment in which the product should be used to preserve its security. For example, in the case of software this may include details of the platform on which it should be installed, any requirements for the trustworthiness and level of skill of potential users, or any assumptions about format or protocol requirements for use with associated software or hardware. For application software, such as a relational database management system (RDBMS), the description of the environment might include details of the product's security dependencies on the base operating system on which it will operate. In the case of hardware, the description of the intended environment might include assumptions about the peripherals to which the product will be connected, e.g. printers and terminals. This is illustrated in Annex 0 and Annex 0.

2.12 The description of the environment could include details of the intended physical environment. For example, the product will be located in a secure area and monitored by trusted administration personnel from a console located in a secure area separate from other users.

2.13 The description of the environment should also include details of any additional measures required to support the product, such as procedural, personnel and physical security measures. For example, in section 2.5 of Annex 0 the possibility of unauthorised users gaining access to the LOCKPC key, is countered by personnel and physical security procedures detailed in the product's user guide. The security target provides accurate references to where this information can be found.

2.14 The intended method of use for the product should be detailed in the product rationale. This will describe the purpose of the product and its features. For example, section 2.1 of Annex 0 describes that the purpose of LOCKPC is to prevent unauthorised access to data, and then describes how it is used to protect separate parts of a PC's hard disk.

2.15 A summary of the security features of the product should be given, which describes the functionality provided to support the security features of the product. This could be a statement defining the functionality class claimed for the product or an overview of the security functionality following a set of generic headings (see below). Any dependencies on other hardware, software or firmware, which are not part of the product itself, should be stated. For example, in the summary of security features section of both Annexes, dependencies are detailed and a statement made that there are no other dependencies which can affect the intended operation of the product. If the product contains security features which are not covered in the security enforcing functions (SEFs) and are not part of the evaluation, these features should either be omitted entirely from the security target, or be clearly distinguished.

2.16 Finally the product rationale should provide an appraisal of the threats to the security of the product from within the environment described. Annexes 0 and 0 provide examples of threats to a product environment under the sub-heading Assumed Threats. Section 2.5 of Annex 0 gives a descriptive assessment of the threats and how they are countered. The approach in Annex 0 has been to describe each threat and show how it is countered by a particular security enforcing function.

2.17 Both Annex 0 and Annex 0 provide product rationales organised with sub-headings for each of the main requirements: intended method of use, intended environment, assumed threats and a summary of the security features.

Security Objectives

2.18 Security objectives are the highest (most abstract) level of security requirements. The ITSEC requires system security targets to include security objectives. The ITSEC does not explicitly require a product rationale to include security objectives, but it is recommended that a product security target includes security objectives as part of the intended method of use.

2.19 Security objectives are derived from the high-level threats to the assets to be protected by the TOE. System security targets will detail actual assets which are to be protected. Product security targets will be less specific, but will identify the type of assets which exist in the product environment (and which are protected by the TOE), such that a prospective purchaser will be able to decide whether the TOE will help to satisfy his system security objectives.

2.20 Specifically, security objectives should identify:

a) the assets which are to be protected in the actual (system) or intended (product) environment

b) the way in which the assets are to be protected, such that the objectives can be related back to the general definition of IT security given in the ITSEC (i.e. confidentiality, integrity and availability).

2.21 It is recommended that individual statements of security objectives are numbered or labelled, to facilitate traceability to the threats and to the security enforcing functions.

Threats

2.22 The ITSEC defines a threat as *an action which might prejudice security*. A security target describes the envisaged threats which may apply to a TOE, and which could cause a TOE to fail to satisfy its security objectives.

2.23 Threats should be stated at a higher level of detail (a lower level of abstraction) than the security objectives. Therefore threats should be more detailed than a simple negation of individual security objectives, since such threats serve little useful purpose. There should be at least one threat for each security objective.

2.24 Specifically, each threat should state:

a) the source of the threat (e.g. a disaffected user or a Trojan horse)

b) what the breach of security is (relating back to the relevant security objective)

c) how the breach of security is achieved (i.e. what the source of the threat does to achieve the breach of security).

2.25 It is recommended that the statement of each threat is labelled to facilitate traceability to the security objectives and countermeasures (i.e. to the SEFs and environmental assumptions).

Specification of the Security Enforcing Functions

2.26 Those features which contribute towards satisfying the security objectives of the product or system are referred to as **security enforcing functions** (SEFs). The SEFs are a key aspect of the evaluation process, and should be defined as clearly and concisely as possible. Taken together the SEFs should give the reader a clear picture of the contribution to security which the TOE can provide. An example of a security enforcing function is a password authentication procedure.

2.27 When defining SEFs it is useful to apply the following checklist:

a) is the meaning clear?

b) is the contribution to security clear?

c) is the SEF implemented by the product or system?

d) is the SEF testable?

e) does the SEF cover only one piece of functionality?

2.28 It is important to establish a stable set of SEFs before the evaluation begins, since changes in this area have a major impact upon evaluation effort and costs.

2.29 There may be other components which, whilst not being security enforcing, must operate correctly in order for security to be maintained. These are **security relevant components**. An example of a security relevant function is a disk subsystem which maintains the integrity of an authorised user and password file.

2.30 An evaluation will concentrate on those components identified as security enforcing and security relevant. However, all other components of the product will be considered and must be shown to be neither security enforcing nor security relevant.

2.31 Privileged components are in general either security relevant or security enforcing. However, particularly in the case of privileged COTS, there may be insufficient documentation available to support the evaluation of a privileged component as part of the TOE. An alternative strategy is to consider the component as security irrelevant, and to justify how the TOE is defended against misuse of the component’s privilege. This may involve any or all of the following:

a) where misuse of the privilege is a threat identified in the security target, the correctness documentation and the suitability analysis should show that the security of the TOE is not compromised in practice

b) where the assignment of privilege to an untrusted component introduces a construction vulnerability, the construction vulnerability analysis should show that the vulnerability is not exploitable in practice, and the binding analysis should be considered in detecting potential vulnerabilities

c) where the assignment of privilege to an untrusted external component introduces a construction vulnerability, the construction vulnerability analysis should show that the vulnerability is not exploitable in practice, the binding analysis should be considered in detecting potential vulnerabilities, and the external interfaces of the untrusted but privileged component should be documented so as to satisfy the requirements of the appropriate assurance level

d) where there is potential for misuse of privilege, the ease of use analysis should show that the misuse is detectable

e) where there is a potential for users maliciously to disregard operational procedures governing the safe use of privilege, the operational vulnerability analysis should show that the vulnerability is not exploitable in practice.

2.32 Where privilege is a relevant issue, the sponsor should show that the privilege used is the minimum required for the purpose (this is known as the principle of least privilege) and that there is no potential for misuse by any other (possibly security irrelevant) process which may inherit the privilege.

2.33 The security target must include a specification of the SEFs provided by the product or system. This specification can be explicit, using a set of generic headings, or it can be made in terms of a reference to one of the predefined functionality classes, or a combination of the two.

2.34 A set of optional generic headings, which can be used for the specification of security enforcing functions, are defined in Chapter 2 of the ITSEC [Reference 0]. These are as follows:

a) Identification and Authentication

b) Access Control

c) Accountability

d) Audit

e) Object Reuse

f) Accuracy

g) Reliability of Service

h) Data Exchange.

2.35 The SEFs specified in section 3 of Annex 0, are written in natural language, providing a statement of the two elements of the product's security functionality. It would be impractical to divide the SEFs by generic headings in this case since there are only two such functions. It is unusual to have as few as two SEFs, but the example product is very simple.

2.36 Example predefined functionality classes are detailed in Annex A of the ITSEC. There are five main classes which have been derived from the functionality requirements of the TCSEC classes [Reference 0]. These are F-C1, F-C2, F-B1, F-B2 and F-B3, intended to be equivalent to the functional aspects of the TCSEC criteria C1, C2, B1, B2 and B3. There is no F-A1 class as the functionality for A1 is unchanged from B3 functionality. In addition, there are other classes, which may be used for products requiring to make claims for more specialist requirements.

2.37 The example at Annex 0 specifies its SEFs using each sentence taken from the description of the predefined functionality class F-C2, contained in Annex A of the ITSEC.

2.38 The UK ITSEC Scheme has expanded the documentation of the ITSEC functionality classes F-C2 and F-B1. This expanded documentation is intended to improve the usability of these functionality classes.

2.39 The SEFs should be correlated to the intended method of use for the product, and to the assumptions about the environment specified in the product rationale. For instance, where weaknesses are recognised in the security functionality there should be a description of the procedural or operational measures used to support that security functionality. For example, the identification and authentication function of a product may be recognised as weak; however, the procedural and physical controls employed in the use of the product may make these weaknesses manageable. This correlation should illustrate any dependencies on other security enforcing functions or non IT security measures provided by the environment.

2.40 At all assurance levels the SEFs must be specified in an informal style, using natural language. In addition to the informal specification, there should be a semi-formal specification at levels E4 and E5. The contents of these two styles of specification should be related so that it is possible to correlate the functionality described in the two interpretations. Examples of semi-formal specifications include the use of graphical representations such as data flow, entity relation and state transition diagrams as implemented by the structured design methodologies, e.g. Yourdon and SSADM. At E6 a formal specification of the security functions and architecture must be provided which must be consistent with the underlying security policy model and informal specifications. At E4 and above a formal model of security policy for the product or system should be included. The security policy model can be a reference to published models, e.g. Bell - La Padula. Alternatively the model could be provided as part of the security target, related to some or all of the security enforcing functions.

Definition of Required Security Mechanisms

2.41 In some cases, for example where a system is mandated to use a particular encryption algorithm, a specification of prescribed security mechanisms may be included in the security target. Similarly, a sponsor may select a predefined functionality class which mandates a prescribed mechanism. A sponsor may also choose to include such mechanisms in the security target, to draw attention to them for commercial reasons.

2.42 The declaration of mechanisms in the security target is optional. However, if they are specified in the security target, the developer must ensure that they are implemented as described.

2.43 Declared mechanisms must be correlated to the security enforcing functions which they are intended to implement. A single mechanism may help to implement many security enforcing functions, while a single security enforcing function may be implemented by the combination of many mechanisms.

2.44 Section 4 of Annex 0, includes a definition of security mechanisms. It describes how each SEF is implemented and correlates the mechanism definition to each SEF.

2.45 The definition of security mechanisms in section 4 of Annex 0 uses the same numbering scheme and function headings as the specification of the SEFs detailed in section 3 of the same Annex. In this way, it is clear which mechanism implements which SEF; this is one method of meeting the ITSEC requirement for traceability.

Claimed Rating of Minimum Strengths of Mechanisms

2.46 The security target must include a claimed rating for the minimum strength of the security mechanisms acting in concert. Each critical mechanism will have been given a rating based on its ability to withstand direct attack. The rating can be either **basic**, **medium** or **high** and, for each mechanism, is based on the level of knowledge and resources that the attacker would need in order to compromise the security of the mechanism.

2.47 The ratings are defined as follows:

a) For the strength of a critical mechanism to be rated **basic** it shall be evident that it provides protection against random accidental subversion, although it may be capable of being defeated by knowledgeable attackers

b) For the strength of a critical mechanism to be rated **medium** it shall be evident that it provides protection against attackers with limited opportunity or resources

c) For the strength of a critical mechanism to be rated **high** it shall be evident that it could only be defeated by attackers possessing a high level of expertise, opportunity and resources, successful attack being judged to be beyond normal practicality.

2.48 The resources required are measured in terms of the time necessary to make a successful attack, whether collusion with a user or administrator is necessary for the attack to succeed, how much expertise is necessary, and whether any specialist equipment is required.

2.49 A basic rating would not be achieved if a successful attack were possible using a low level of all of these resources, for example if the attacker was a layman who was successful in minutes, used no equipment and needed no assistance. A mechanism would achieve a basic rating if the layman needed days of effort, or if it took a knowledgeable individual minutes to succeed.

2.50 Conversely a high strength would be achieved if the attacker required assistance from an administrator, or if the attacker needed months of work using specialist equipment. A medium rating would fall somewhere between the two. A more comprehensive explanation of this issue can be found in the ITSEM [Reference 0].

2.51 An overall rating for the product is derived by considering the strengths of all mechanisms, together with the possible modes of attack. Along any attack path containing a number of countermeasures there must be an individual mechanism which has a greater or equal strength to that claimed overall. Otherwise, the attack will represent an exploitable vulnerability. For example, in Annex 0 the claimed strength of mechanism is medium. Therefore, the strength of at least one mechanism along each attack path must be either medium or high.

2.52 A statement of the kind shown in section 5 of Annex 0 is sufficient for the specification of the claimed strength of mechanism. A detailed analysis of the rationale behind the rating, using the criteria specified above, does not need to be included in the security target, but should be provided as part of the evaluation deliverables, probably as a separate document (the strength of mechanisms analysis). A similar statement is shown in Annex 0.

2.53 Special considerations for the rating of the minimum strength of cryptographic mechanisms are given in the section starting at paragraph 0.

Target Evaluation Level

2.54 The security target must specify a target evaluation level for the product or system. This should be in the form of a simple statement as illustrated in section 6 of Annexes 0 and 0. The various possible target evaluation levels represent ascending levels of assurance, defined as confidence in the correctness and effectiveness of the implementation of the product's security enforcing functions and mechanisms.

2.55 The requirements for each assurance level are outlined below and detailed in Chapter 4 of the ITSEC. The requirements at each level build on those of the previous level.

a) **E0** - represents inadequate assurance and cannot be claimed for any TOE undergoing evaluation, as the TOE may only achieve E0 as a result of an evaluation.

b) **E1** - requires a security target and an informal description of the architectural design of the TOE. Functional testing should be performed to show that the TOE satisfies the security target. User and administration documentation must give guidelines on maintaining product security. In addition, the TOE must be uniquely identified and have delivery, configuration, startup and operational documentation. There must be evidence that secure distribution methods have been utilised.

c) **E2** - requires an informal description of the detailed design and test documentation. The separation of security enforcing and other components must be shown within the architectural design. The developer's configuration control procedures, and security measures adopted to maintain the integrity of the product, will be assessed. Audit trail output, if produced during startup and operation, is required.

d) **E3** - requires source code or hardware drawings in relation to SEFs and security relevant functions, with an informal description of the correspondence from these to the detailed design. There must be evidence that acceptance procedures are used and that retesting has occurred after the correction of errors. The implementation languages used should conform to recognised standards.

e) **E4** - requires a formal model of the TOE's security policy along with a semi-formal specification of the SEFs, architectural and detailed design. There must be evidence that testing covers all SEFs in sufficient detail. The TOE, and any tools used, must be under configuration control with any changes being audited. All compiler options should be documented. The TOE must retain its security after restart as a result of failure.

f) **E5** - requires the architectural design to explain the interrelationship between security enforcing components, with close correspondence between the detailed design and source code/hardware drawings. There must be information on the integration process and run time libraries. Configuration control must be independent of the developer. Configured items must be identified as security enforcing or security relevant, with support for variable relationships between them.

g) **E6** - requires a formal description of the architecture and SEFs. There must be correspondence from the formal specification of the SEFs through to source code and tests. Different TOE configurations shall be defined in terms of the formal architectural design. All tools shall be subject to configuration control.

Scope of Evaluation

2.56 It is important to ensure that the scope of the TOE is clear, and that the threats which are identified are completely countered by the defined SEFs and any operational assumptions. This is particularly important where the TOE relies upon other products to help enforce security (e.g. a secure database), as the limits of the evaluation process in such cases must be clearly understood.

2.57 The version of the TOE, the platforms and the evaluated configuration should also be clearly defined.

Chapter 3 Writing Correctness Documentation

Introduction

3.1 This chapter illustrates the basic requirements and considerations when writing correctness documentation in support of an ITSEC evaluation. The chapter concentrates mainly upon product oriented correctness documentation although it is also generally applicable for systems.

3.2 Evaluation using the ITSEC requires the separation of an IT system or product into the areas of functionality and assurance.

3.3 Functionality specifies the security enforcing functions contained within the TOE.

3.4 Assurance is further split into assurance (confidence) in the correctness of the security enforcing functions of the TOE, and assurance (confidence) in the effectiveness of those functions. This chapter concentrates on correctness requirements. (Effectiveness requirements are considered in the following chapter.)

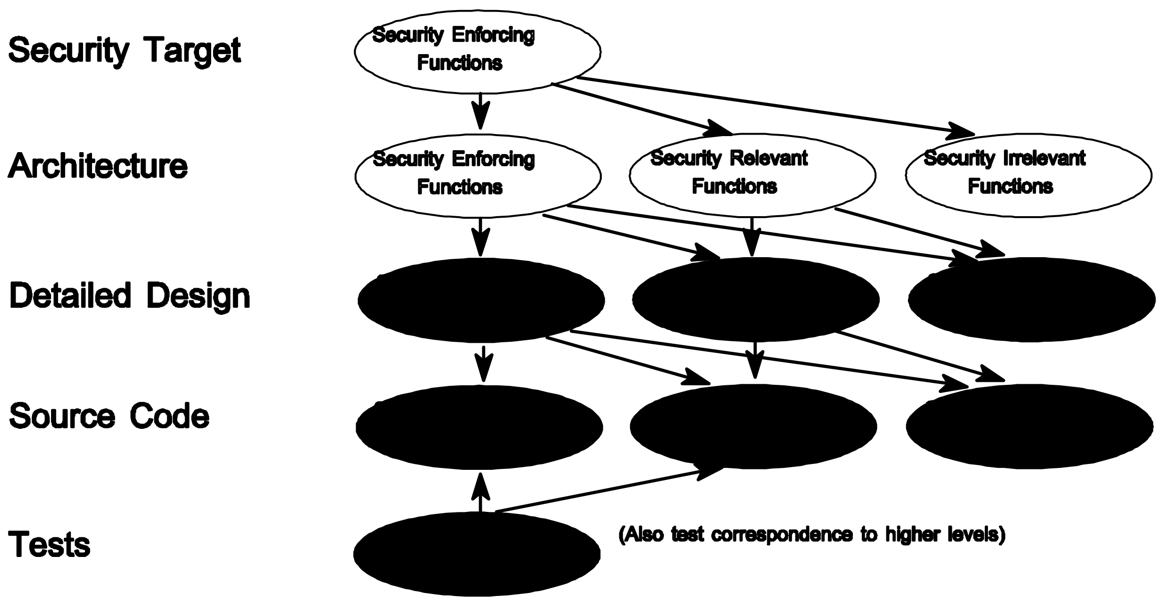
Aim

3.5 Correctness is concerned with ensuring that the TOE accurately implements the security target, and that no errors have occurred during the refinement and implementation of the design.

3.6 The development of any TOE involves several stages of refinement and integration, with representations of the TOE at each level of abstraction. The security target provides the highest level of abstraction defining the TOE. The operational TOE, in the form of executable code or electronic circuitry, provides the most concrete and detailed representation. The levels of representation grow gradually more detailed, providing a greater refinement of the security objectives. Thus, the detailed design is a refinement of the architectural design, which is a refinement of the security target.

3.7 The description of a security function at any given representation is said to be a correct refinement if the totality of effects at that representational level exhibits all the effects described at the preceding (higher) representational level and no others.

3.8 The purpose of the ITSEC correctness criteria is to help establish the fact that each representation is a correct refinement of its corresponding higher level representation, and that no errors have been introduced. The mapping between the security target and the TOE is provided by mappings between intermediate representations. This is illustrated in Figure 0 below.


Security Target
Security Enforcing Functions
Architecture
Security Enforcing Functions
Security Relevant Functions
Security Irrelevant Functions
Detailed Design
Source Code
Tests
(Also test correspondence to higher levels)

Note: Not all deliverables are required at all levels.

Figure 0 Refinements of correctness

Correctness Documentation

Approach

3.9 The following documentation is required for an ITSEC evaluation:

a) Development Process (TOE representations):

- security target
- for E4 and above, formal security policy model
- architectural design
- for E2 and above, detailed design
- for E3 and above, source code or hardware drawings
- for E2 and above, test documentation

b) Development Environment:

- configuration list
- for E2 and above, configuration control documentation
- for E3 and above, details of programming languages and compilers
- for E2 and above, details of developer's security

c) Operational Documentation:

- user manuals
- administration manuals

d) Operational Environment

- delivery and configuration documentation
- startup and operation documentation.

3.10 The ITSEC criteria do not prescribe any particular development method, and do not assume that the TOE documentation will be produced in any particular order. However, the evaluators will expect to examine the Development Process documentation hierarchically, so that they can confirm the correct refinement of the security functionality through the various representations. The other correctness deliverables (listed above) can be provided and evaluated as and when convenient (but always after the security target).

3.11 The requirements for traceability between the TOE representations are covered in a separate chapter of this part of this Guide, as are the requirements for the security target.

TOE Representations

3.12 The requirements at each representational level may be satisfied by a single document or by a number of documents in combination.

3.13 For a small TOE, it is possible that various chapters in a single document can satisfy all the requirements for the development process. For a larger TOE, it may still be a good idea to produce a single document pointing to where the ITSEC requirements are met in the TOE documentation.

3.14 For most well organised developments, architectural and detailed design documentation will be produced as a matter of course, and should cover many of the ITSEC requirements. One aspect which is likely to need additional attention is the provision of information showing how the security functionality is provided within each level of design; this aspect is considered in the *Traceability* chapter elsewhere in this part of this Guide.

3.15 For evaluations at E5 and above, the detailed design is required to use *layering, abstraction and data hiding*. The ITSEM explains that these techniques are intended to aid understanding and maintenance of the design, so that each level of design contains the right amount of information to be understood, hiding design complexities within successively lower levels.

3.16 Also for the higher level evaluations, the detailed design should include all global variables, and all variables shared between groups of functions, in the consideration of interfaces. The interface specifications must be consistent and complete, as they will be checked against the design of the security enforcing and security relevant functions.

3.17 Source code or hardware drawings will exist for every TOE, though it need only be provided for evaluations to E3 and above. However, the release to the evaluators of any third party software may need to be considered.

3.18 There may be circumstances where source code or hardware drawings will be required for evaluations at E1 or E2. This may be where, for example, a high strength of mechanism is being claimed (at E1 the detailed design for the high strength mechanism(s) may also be needed), or where there are additional requirements on the evaluators exceeding the ITSEC criteria, such as for evaluations of cryptographic equipment for HMG use.

3.19 Developers should also take care that the comments within the source code provide assistance to the evaluators in understanding the code. The comments should describe the way the code currently works, not the way it used to work, and should not consist solely of unhelpful comments such as "zap it", or "this might work".

3.20 As with the architectural and detailed design documentation, traceability of the security enforcing functions and mechanisms in the source code or hardware drawings is likely to require additional information to that normally produced during the implementation process.

3.21 When providing test documentation, developers should be aware that the evaluators will require to repeat a sample of the tests, to ensure that the tests work as described and that the documented results are obtained. For this reason, a test suite that has been hanging around at the back of the cupboard is not a good candidate for supply to the evaluators unless there is a high degree of confidence that it will perform the desired functions and return the expected results. It is also likely that this will not meet the requirement for regression testing following correction of errors.

3.22 To meet the requirement for regression testing, at E3 and above, the developer should aim to show that, not only has the correction itself been tested and found to work correctly, but also that a strategy has been used to test at least some portion of the rest of the system to ensure that no other problems have been introduced. The evaluators will check that the stated strategy for regression testing has been followed, by sampling evidence related to corrected faults.

3.23 Prior to the evaluators' visit to repeat the developer's tests, it is recommended that a trial run of the test suite be performed, to ensure that everything will go smoothly on the day.

3.24 The test documentation should be written in such a manner that the evaluators will be able to run a sample of tests with little support. It should not be based on assumptions about the environment, the configuration, or the test engineer's skills and experience (where specific configuration, skill or experience is needed, the requirement should be documented). For example, test suites that include statements such as "run Joe's restart test" are not a good idea, unless Joe's restart test is explicitly documented.

3.25 The test documentation should clearly identify the purpose of each test, so that the evaluators can identify the objective. It should also identify the related SEFs for each test (this will assist in meeting the traceability requirements). The evaluators will check that "for each testable statement in the security target, at least one test has been defined to demonstrate the statement" (ITSEM 4.5.70).

3.26 For the higher assurance levels, the tests must cover the security functionality in more detail. At E3 and above, the tests must cover every interface to every security enforcing or security relevant basic component identified in the detailed design and every security mechanism in the source code or hardware drawings. It is important to note the requirement for test coverage of security relevant aspects here; the greater the exclusion of unnecessary functionality from the security enforcing and security relevant functions, the easier it will be to meet this test requirement. Thus, care taken over the specification of the detailed design will reap benefits at the testing stage.

Development Environment

Configuration List

3.27 The configuration list must identify the version of the TOE under evaluation. For evaluations at E2 and above, it must also identify the components (and their versions) which comprise the TOE, including design documentation, make files, source code, operational manuals, release notes, and tools.

Configuration Control

3.28 An evaluation to E2 and above, requires that the TOE is under configuration control. Configuration control can be effected by a combination of measures, though for E4 and above it must be mainly tool based. The controls on who can accept items into configuration control and authorisation for changes become more stringent with assurance level. The objective of the configuration control system is to be able to explicitly identify the components of the TOE, so that a fault reported in an older version can be investigated (see also the requirements for *Delivery and Configuration* in the *Operational Environment*). It should therefore be possible to recreate a specific version of the TOE, and identify the documentation associated with that version.

3.29 For evaluations to E5 and E6, it must be possible to identify all items affected by a change and whether they are security enforcing or security relevant. It may not be possible to fully automate this process, and the initial identification of each item as security enforcing or security relevant, together with the relationships between items, may have to be established manually.

3.30 For evaluations to E2 and above, the TOE, its components and documentation must all possess a unique identifier, which should be used in all references. The source code and hardware drawings must also be uniquely identified for E3 and above.

Programming Languages and Compilers

3.31 Evaluations to E3 and above require that the languages and compilers used:

a) be well-defined, so that the meaning of all source code statements is unambiguous

b) have documented implementation dependent options; for E4 details of the selected compiler options should also be provided.

3.32 In addition, for evaluations to E6, the source code of any run time libraries must also be provided.

Developer's Security

3.33 Evaluations to E2 and above require the following information regarding the security of the development environment:

a) intended protection for the integrity of the TOE and the confidentiality of the associated documentation

b) physical, procedural, personnel and other security measures.

Operational Documentation

3.34 User manuals must be provided for evaluations at all levels, and must have the following characteristics:

a) cover the security enforcing functions relevant to the end user

b) contain guidelines for secure operation, showing how to use the TOE in a secure manner

c) be structured, internally consistent, and consistent with other TOE documentation.

3.35 Administration manuals must be provided for evaluations at all levels, and must have the following characteristics:

a) cover the security enforcing functions relevant to the administrator

b) cover the functions which provide information, and those which control security parameters, covering all parameters under the administrator's control

c) cover each type of security relevant event

d) contain details of administration procedures, showing how the TOE is administered in a secure manner

e) contain guidelines on the consistent and effective use of the TOE's security features, and how these interact

f) contain instructions for installing and configuring the TOE

g) be structured, internally consistent, and consistent with other TOE documentation.

Operational Environment

Delivery and Configuration

3.36 The delivery and configuration procedures must be identified for evaluations at all levels, showing how they maintain security, and covering:

a) the impact of different configurations on security

b) the procedures for delivery and system generation

c) for E2-6, the delivery procedure must guarantee the authenticity of the delivered TOE, and must be approved by the national certification body

d) for E2-6, generation of the TOE must be audited such that it is possible to reconstruct how and when the TOE was generated.

Delivery Procedures Approved by the UK Certification Body

3.37 The Certification Body has issued the following guidance for delivery procedures for TOEs evaluated under the UK Scheme.

3.38 **At E1 a delivery procedure must be documented**.

3.39 At E2 a delivery procedure must be documented **and applied. The procedure should state a method available to the receiver to detect interference with the TOE during delivery.**

3.40 At E3 a delivery procedure must be documented and applied. The procedure should **describe** methods available to the receiver to:

a) detect interference with the TOE during delivery,

b) **detect that an unauthorised agent has initiated delivery**.

3.41 At E4 a delivery procedure must be documented and applied. The procedure should describe methods available to the receiver to:

a) detect interference with the TOE during delivery,

b) detect that an unauthorised agent has initiated delivery,

c) **detect any modification to the TOE during delivery**.

3.42 At E5 a delivery procedure must be documented and applied. The procedure should **explain** methods available to the receiver to:

a) detect interference with the TOE during delivery,

b) detect that an unauthorised agent has initiated delivery,

c) detect any modification to the TOE during delivery.

3.43 **At E5, the procedure should explain why the delivery path is trusted (trusted transfer).**

3.44 At E6 a delivery procedure must be documented and applied. The procedure should explain methods available to the receiver to:

a) detect interference with the TOE during delivery,

b) detect that an unauthorised agent has initiated delivery,

c) detect any modification to the TOE during delivery.

3.45 At E6, the procedure should explain why the delivery path is trusted (trusted transfer). **It should explain a method available to the originator and receiver to authenticate the TOE**.

3.46 Where company standard delivery procedures and practices are suitable, these should be utilised to their full potential. However accreditation to a recognised standard, e.g. BS EN ISO 9000, may not necessarily satisfy all the demands stated in these requirements

3.47 Where protective security requirements apply, they must be applied in addition to the above requirements. Where a conflict arises between these stated requirements and those concerned with protective security then the latter should take precedence once endorsed by the certifier.

Startup and Operation

3.48 Procedures for secure startup and operation must be detailed for evaluations at all levels, showing how they maintain security, and covering:

a) for E2-6, details of any security enforcing functions that can be deactivated or modified during startup, normal operation or maintenance

b) for E4-6, procedures to restore the TOE to a secure state after failure or software/hardware error

c) for E2-6, diagnostic tests (administrator, end-user or automatically initiated) for any security enforcing hardware components, these tests being available in the operational environment.

3.49 In addition, for evaluations at level E2 and above, the following evidence must be provided:

a) example results from all diagnostic tests for security enforcing hardware components

b) example audit trail output during startup and operation.

Chapter 4 Writing Effectiveness Documentation

Introduction

4.1 This chapter illustrates the basic requirements and considerations when writing security effectiveness documentation in support of an ITSEC evaluation. The chapter concentrates mainly upon product oriented effectiveness documentation although it is generally applicable to system effectiveness documentation.

4.2 No prior exposure to the ITSEC or the ITSEM is assumed in this chapter.

Aim

4.3 Effectiveness is concerned with ensuring that: security enforcing functions (SEFs) protect the specified assets from the threats specified in the security target (suitability of functionality); the individual security enforcing functions will be secure against the security target when considered as a whole (binding of functionality); there are no exploitable vulnerabilities in the system or product when considered as a whole with its operational environment (vulnerability assessments, strength of mechanisms, and ease of use).

4.4 The ITSEC defines effectiveness evaluation in paragraphs 1.14 and 1.15 as:

"Evaluation of effectiveness assesses whether the security enforcing functions and mechanisms that are provided in the TOE will actually satisfy the stated security objectives. The TOE is assessed for suitability of functionality, binding of functionality (whether the chosen functions work together synergistically), the consequences of known and discovered vulnerabilities (both in the construction of the TOE and the way it will be used in live operation), and ease of use. In addition, evaluation of effectiveness assesses the ability of the security mechanisms of the TOE to withstand direct attack (strength of mechanisms)."

4.5 Effectiveness can be summarised as answering the question "are the security measures implemented in the system or product effective against the threats identified in the security target and free from exploitable vulnerabilities?"

4.6 The division between correctness and effectiveness is partly dependent upon the detail contained in the security target. As described in the previous chapter, correctness concerns the correct implementation of the security functionality specified in the security target. Effectiveness, on the other hand, is concerned with ensuring that there are no exploitable vulnerabilities in the operational TOE.

4.7 For example, the security target may identify a high strength mechanism for authenticating users of a system. If the implementation used a two-character password then the TOE would fail on effectiveness grounds, since a two-character password could easily be guessed by trial and error. If, however, the security target had also specified that a ten-character password be used, then an implementation which allowed a two-character password would fail on correctness grounds. In either instance the evaluation team would identify a failing in the TOE for which the sponsor would need to identify a remedy.

4.8 As the assurance level increases, so the requirement for detail increases. This extra detail is used in an increasingly more rigorous correctness assessment.

4.9 The requirements for effectiveness do not change as the assurance level increases, but instead build upon the increasing correctness requirements. The refinement of detail at the lower levels of abstraction often introduces extra sources of potential vulnerabilities which must be addressed by the sponsor.

4.10 During the correctness aspects of the ITSEC evaluation, the evaluators will build up an understanding of the TOE which will be used during vulnerability analysis to identify any weaknesses. As a result, although the correctness and effectiveness aspects can proceed together, the effectiveness aspects can only be completed after the correctness aspects. Any errors found during the correctness aspects will be used as a source of possible effectiveness vulnerabilities.

Effectiveness Documentation

Approach

4.11 For all ITSEC assurance levels the following set of effectiveness documentation is required:

- suitability analysis,
- binding analysis,
- strength of mechanisms analysis,
- construction vulnerability analysis,
- operational vulnerability analysis,
- ease of use analysis.

4.12 Effectiveness documentation should be produced in conjunction with the development of the TOE, although it can be retrospectively produced for TOEs which are undergoing subsequent evaluation. However, retrospective production can present significant problems for ensuring compliant effectiveness documentation, and should be avoided. It is also conceivable that this process could identify a vulnerability for which there is no counter.

4.13 Whilst it may be possible to begin production of some of the effectiveness documents once the security target has been established, it will not be possible to complete the documents at the beginning of the development. The effectiveness documentation for a TOE will evolve with the development of the TOE. As the TOE functionality, architecture, design, implementation, and test phases are conducted, so the effectiveness documents will require revision or addition to reflect the current state of the TOE development. This implies that the document production must be an iterative process. At each stage of the development, the vulnerability assessment should be repeated until all of the information relevant to the assurance level has been considered.

4.14 Figure 2 illustrates the principal order in which to approach the production of the effectiveness documents. Note that in practice the style and content of each effectiveness document will depend greatly upon the nature of the TOE and only general guiding principles are given in this chapter. The sponsor may choose to provide each document separately or to combine them.

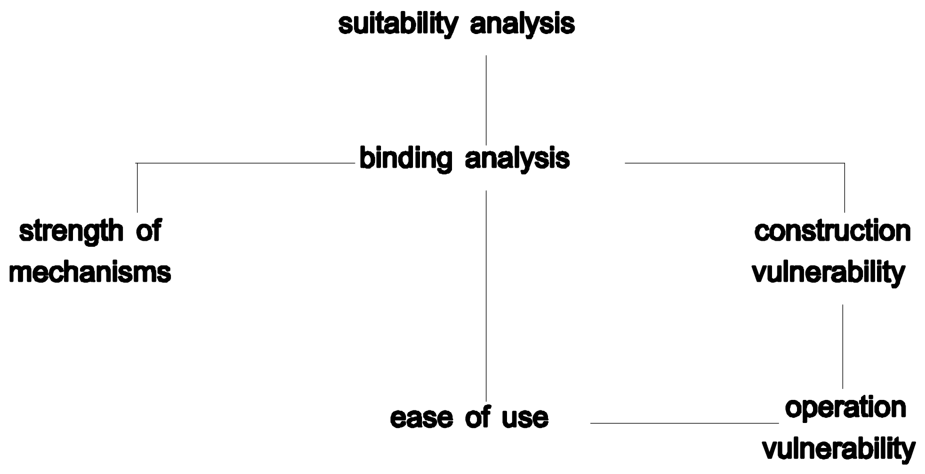


Figure 2 Outline approach to the production of effectiveness documentation

4.15 Note that the effectiveness analyses, except the suitability analysis, should consider all of the correctness documents identified in ITSEC Figure 4 for the target assurance level; the suitability analysis may refer only to the security target. While this means that the effectiveness analyses cannot be fully completed until the relevant correctness documents have been produced, much of the analyses can be undertaken in advance, with the benefit that any shortcomings in the deliverables discovered during the analyses can be quickly corrected.

4.16 In general, the first document to be produced after the security target should be the suitability analysis. This should link the SEFs and mechanisms defined in the security target to the threats that have been identified in the security target, and show how all of the threats are countered.

4.17 Following this, the binding analysis can be produced. This should identify all potential interrelationships between the SEFs and mechanisms. It should show that for all identified relationships, no conflicts that would compromise the effectiveness of the SEFs can occur. This demonstrates that the SEFs and mechanisms work together in a way that is mutually supportive and provides an integrated and effective whole.

4.18 It is then possible to produce the strength of mechanisms analysis, the construction vulnerability analysis, and the operational vulnerability analysis concurrently. The strength of mechanisms analysis should identify all the critical security enforcing mechanisms of the product or system and demonstrate that each mechanism can withstand a direct attack up to the claimed strength of the mechanism. Analysis of the underlying algorithms should be included, with an assessment of the level of expertise and resources necessary for a successful attack.

4.19 The construction vulnerability analysis should identify any vulnerabilities regarding the construction of the product or system known to the sponsor (for example, failing to clear a buffer). These can include ways of deactivating, bypassing, corrupting, or circumventing the SEFs and mechanisms (all failures of binding). Each vulnerability should be assessed for its usefulness in attacking the security of the product or system in its operational environment. The countermeasures or corrective actions proposed to counter the vulnerability should be stated with supporting arguments as to why they should be effective.

4.20 The operational vulnerability analysis is similar to the construction vulnerability analysis, but concentrates upon vulnerabilities in the way in which the product or system is used. Operational vulnerabilities take advantage of weaknesses in non-technical countermeasures to violate the security of the system or product (e.g. the unauthorised disclosure of a password to another user). They relate to the boundary between IT and non-IT countermeasures, such as operational procedures and physical security. They will be considered during the evaluation if they appear as part of the operational documentation or as part of the product rationale or SSP. Evaluators will be concerned with ensuring that non-IT countermeasures are effective against known constructional vulnerabilities.

4.21 Before entering an evaluation the sponsor must seek guidance from the Certification Body on any vulnerabilities which are known to exist in the TOE. There is a standard form for requesting this information; copies are available from the Certification Body. The Certification Body will normally respond to a request for vulnerability information within 10 working days.

4.22 Vulnerability information requests may be made at any time. In particular, requests for vulnerability information may be made for the purpose of providing estimates before formal evaluation starts.

4.23 The sponsor's vulnerability analyses should address any vulnerabilities reported by the Certification Body in response to such a request. The evaluators will ensure that this has been done by checking against the Certification Body's vulnerability database.

4.24 Finally, the ease of use analysis can be produced. This should investigate whether a product or system can be configured or used in a manner which is insecure, but which an administrator or end-user would reasonably believe to still be secure. All intentional and unintentional means of disabling or deactivating the SEFs should be shown to be detectable. The operation of the product or system following an error should also be addressed. It should also be possible to configure and use the product or system in a secure manner using only the information contained in the user and administrator manuals.

4.25 During the course of the evaluation, a number of problem reports may be raised and made available to the sponsor or developer. The effectiveness documentation may be affected by the outcome of these problem reports. The reports may be raised after examination of deliverables for particular effectiveness aspects, or as the result of penetration testing. A TOE will fail evaluation on effectiveness grounds if an exploitable vulnerability is found and is not addressed before the end of the evaluation. Each vulnerability should be eliminated; failing this, the introduction of other countermeasures may be acceptable.

4.26 The evaluation is always conducted with respect to the security target. Thus, if the security target has no requirements for, say, confidentiality of data and a known method of accessing the data exists, then the product or system under evaluation will not necessarily fail on effectiveness grounds, although comment is likely to be made in the certification report.

Document Interdependencies

4.27 Figure 2 illustrates the most likely sources of information for the production of each effectiveness document. It is not necessarily complete and it is up to the author of each particular document to ensure that all relevant sources of information have been considered. The solid arrows indicate a strong (or mandatory) source of information. The outline arrows indicate further potential sources of information. Effectiveness documents are shown in boxes, other documentation is shown in ovals.

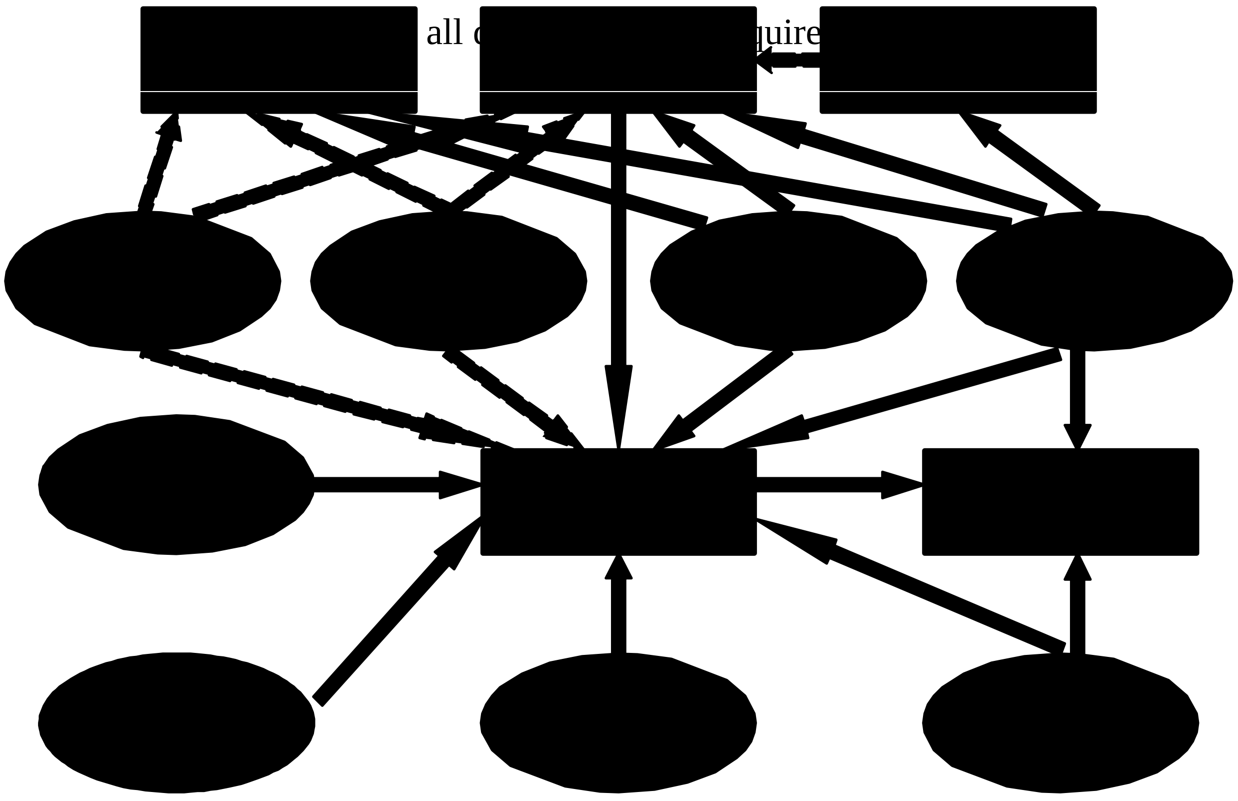


Figure 2 Potential sources of information for effectiveness documents

4.28 The security target is fundamental to the production of the effectiveness documentation since it lists the intended use of the product or system, the operational environment, the assumed threats, the SEFs, the security mechanisms, the minimum strength of each mechanism, and the assurance level.

4.29 Also fundamental to most effectiveness documents is the architectural design. This provides the overall top level definition and design of the product or system, identifying the basic product or system structure, external interfaces, and the implementation of security enforcing functions. At level E2 and above, the separation of security relevant and other components is included.

4.30 As the assurance level increases, more documentation is required; a detailed design document is required for E2, and source code or hardware drawings are required for E3. These documents may also provide input to the effectiveness documentation, especially as the more detailed levels of abstraction contained within these documents may well introduce extra detail concerning the SEFs and identifying further sources of vulnerabilities. For example, the introduction of a variable within the detailed design may identify a possible exploitable vulnerability. Test results and evaluator reports may also provide sources of possible vulnerabilities.

4.31 The developer's register of fault reports is an important input to the vulnerability analysis.

4.32 The user and administration documentation should also provide sources of justification for the claims made in the vulnerability analyses and ease of use documents.

4.33 The following sections describe each effectiveness document in more detail and suggest example document structures together with observations about producing the documents.

Suitability Analysis

Description

4.34 The suitability analysis addresses the suitability of the TOE's SEFs to counter the threats to the security of the TOE as identified in the security target. The suitability analysis should link the SEFs and mechanisms to the threats that they are designed to counter. It should show how the threats are countered by the SEFs and mechanisms. It should show that there are no threats that are not adequately countered by one or more of the stated SEFs. This may expose vulnerabilities arising because a SEF fails to uphold a security objective for a threat identified in the security target.

Approach

4.35 A suitability analysis document could contain a completeness section and an adequacy section within it. The completeness section would link the threats identified in the security target to the SEFs that counter them. This may be accomplished via the use of a table as illustrated in Figure 4 below. This table lists three threats and indicates which SEFs counter them by an asterisk entry in the table. It should either be stated that each threat is only countered if all the environmental constraints given in the security target are met, or the environmental constraints necessary to allow the SEFs to be effective should be listed for each threat.

	Function 1	Function 2	Function 3	Function 4	Function 5	Function 6
Threat 1	*	*				
Threat 2		*			*	*
Threat 3			*	*		*

Figure 4 Example mapping of security enforcing functions to threats

4.36 The adequacy section would show how each threat is adequately countered by the identified SEFs in the security target. The threats can be broken down into different scenarios and then the effect of the relevant SEFs demonstrated.

Considerations

4.37 In preparing for production of the suitability analysis, consideration should be given to the level of detail of the SEFs provided in the security target. These should contain sufficient information to enable the evaluators to complete the analysis based purely on the security target. In addition to SEFs the suitability analysis is required to consider mechanisms. As the specification of these is optional within the security target, the sponsor may choose to refer to information in the architectural design in order to provide clarification.

4.38 The suitability analysis may also consider whether the SEFs support and do not contradict environmental constraints against the configuration and use of the TOE, allowing the TOE to be used as intended. For example, an environmental constraint stipulating that no user should have write privilege would be compromised by a SEF stating that all files must be owned by a user and owners of files have write privilege. If the system or product can be configured insecurely, then the reasonableness of any environmental constraints which would still enable the security target to be met might also be considered in the suitability analysis. If the suitability analysis does not address the effects of environmental constraints upon the security functionality then these could be considered in the vulnerability analysis instead.

4.39 The requirement to show that the SEFs counter the threats is based on the premise that the statements of threat (in the security target) are more detailed that the statements of security objectives. In this case, there should be sufficient detail of the threats to perform the suitability analysis, possibly with minimal amplification, such as picturing different threat scenarios. Where the statements of threats leads to more detailed statements of security objectives, then these security objectives should be drawn into the suitability analysis.

Binding Analysis

Introduction

4.40 The system or product will operate securely if the SEFs:

a) counter the threats to the security of the TOE

b) are not bypassed, nor their intent contradicted

c) are implemented correctly, by mechanisms of sufficient strength.

4.41 The binding analysis addresses the issue of indirect attack against the TOE (bypassing or contradicting the SEFs), while the suitability analysis addresses the suitability of the TOE's SEFs to directly counter the threats to the security of the TOE.

4.42 The binding analysis assesses the ability of the TOE's SEFs and mechanisms to bind together in a way that is mutually supportive and provides an integrated and effective whole. The binding analysis should provide an analysis of all potential interrelationships between SEFs and mechanisms and show that it is not possible to cause any SEFs or mechanisms to conflict with, or contradict the intent of, other SEFs or mechanisms.

4.43 Any error or inconsistency found at any point during the binding analysis would constitute a construction vulnerability.

The Context of Binding

4.44 The construction correctness analyses check the correct realisation of individual security enforcing functions from their specification in the security target to their implementation in code or hardware. This traceability activity is complemented by four construction effectiveness analyses:

a) The suitability analysis checks that the set of security enforcing functions presented are both necessary and sufficient to counter the identified threats.

b) The binding analysis checks that the realisations, within the TOE, of the security enforcing functions are able to withstand indirect attack by providing a secure, integrated whole.

c) The strength of mechanisms analysis checks the ability to withstand direct attack of those mechanisms which, because of the essential nature of their conception, are vulnerable to such attack.

d) The construction vulnerability analysis assesses the exploitability of potential construction vulnerabilities, involving those identified in earlier construction work packages, in addition to any reported by the sponsor or suggested by the Certification Body.

4.45 Binding analysis is thus seen to be concerned with vulnerabilities in the construction of the TOE which, if exploited by indirect attack, would prevent the TOE from meeting one or more of its security objectives. If the TOE is to successfully defend itself against indirect attack then its security functions must support each other where necessary; also the security functions must not conflict with each other. In practice this involves consideration of the potential interactions, between security functions, which are provided for and permitted by the construction of the TOE.

4.46 For example, consider a pair of security functions: an Identification and Authentication function which requires the secure storage of authentication data; and a DAC function which requires the access which subjects have to secure data objects to be controlled. If the DAC function is selected to ensure the secure storage of the authentication data then it is seen to support the Identification and Authentication function, and this interaction is required to enforce binding. In order to ensure that the binding is not invalidated, the possibility of other interactions with the stored authentication data, involving circumvention of the DAC support, must be excluded.

Bases for Binding

4.47 ITSEC 3.17-3.19 indicates that binding analysis is concerned with 'security enforcing functions and mechanisms'. However binding must also be demonstrated for lower levels of representation than that of the security enforcing functions, in accordance with the requirements of the target assurance level as given by ITSEC Figure 4.

4.48 Note that binding is undertaken on the assumption that all security enforcing functions are correctly realised. Therefore, correct binding of the security enforcing functions will automatically provide a degree of assurance for the correct binding of the lower level representations.

4.49 It must be recognised however that, for complete two-way traceability of security enforcing functions to exist, not only must the full security functionality exist at all levels of representation, but additional functionality (which could invalidate higher level binding) must not be introduced at lower levels. Note that, whilst this becomes increasingly significant at higher assurance levels, it is in part consistent with stricter design criteria such as "It shall be structured into well-defined, largely independent basic components" (ITSEC E4.8) and "Unnecessary functionality shall be excluded from security enforcing and security relevant components" (ITSEC E5.8). Realistically however, additional detail of potential significance to binding can be expected to exist at lower representational levels and must be considered.

4.50 At assurance levels E3 and above it is considered that the mechanisms identified during correctness offer the clearest basis for a binding analysis. This is reflected in the following subsection, which contains a full discussion of the mechanism basis. Alternative approaches for these assurance levels are then discussed briefly. Finally appropriate bases for assurance levels E1 and E2 are considered.

Mechanism-based Binding - E3 and above

4.51 The appropriateness of the mechanisms identified during correctness as a basis for a binding analysis takes account of the following considerations:

a) Essentially, security functions and mechanisms are equivalent, mechanisms providing the abstract implementation of security functions. Consideration of mechanisms will therefore be sufficient to meet the ITSEC criteria which require consideration of 'security enforcing functions and mechanisms'.

b) Certain mechanisms which require consideration in binding will exist at a lower level of representation than that of typical security enforcing functions, e.g. a protection mechanism offered by underlying hardware which is identified in the Architectural Design. Consideration of security enforcing functions alone may therefore be insufficient to meet those ITSEC criteria which require consideration of 'security enforcing functions and mechanisms'.

c) Mechanisms offer a better basis than security functions for traceability to low level representations, by virtue of including key implementational concepts (i.e. key secure data abstractions and high level security logic).

d) The mechanisms can together be expected to provide a clear, accessible, logical, model of the TOE, which remains uncluttered by virtue of excluding implementational constraints.

4.52 In many cases a mechanism-based binding analysis will need to be supplemented by significant detail from lower level representations (i.e. from low level physical design components identified in detailed design and from implementation components), in accordance with the target assurance level as given by ITSEC Figure 4. This will be necessary in the following respects:

a) To confirm that binding of a generic object type holds for all object instantiations.

This concerns the full traceability of binding interactions.

For example, consider the binding, within a single ACL mechanism, of a number of DAC security enforcing functions (see below for a consideration of the binding of security enforcing functions within a mechanism), where the mechanism and security enforcing functions are concerned with access control on a generic 'object'. It is necessary to confirm, from lower representation levels, that this binding holds for all object types, e.g. files and database tables.

b) To confirm that binding of a generic data class holds for all data store instantiations.

Again this concerns the full traceability of binding interactions.

For example, an access control mechanism may support audit by applying access control to stored audit information. The implementation is such that the audit information is held in a set of audit data files, and it is necessary to confirm that the access controls are applied to each file.

c) To confirm that binding of an abstract data item holds for the set of its implementational attributes.

Again this concerns the full traceability of binding interactions.

For example a mechanism, available to the TOE administrator, to create and delete user groups might support an access control mechanism which controls object access capabilities assigned to group subjects. A binding vulnerability will be apparent from a lower level of representation if deletion of a group does not include deletion of the group Id, which remains associated with established access permissions. If it is then possible to create another group having the same group Id, then the access permissions intended only for the original group will automatically transfer to the new group.

d) To confirm that the binding is not invalidated by low level effects.

This concerns the possibility of incomplete two way traceability.

For example if access to secure data is policed by a file access control mechanism, it might be necessary to confirm that this could not be circumvented by a lower level disk block access capability.

e) To confirm that the binding is not invalidated by low level components which implement high level interactions.

This is a special case of ensuring that binding is not invalidated by low level effects. It applies principally to external interface components (see below under *External Interface Considerations*) and reference monitors (see below under *Techniques to enable a Pragmatic Approach to Binding*).

4.53 Where a given mechanism implements a number of security enforcing functions then the binding, within the mechanism, of the security enforcing functions must be considered. At the mechanism level this can be achieved by confirming that the mechanism is integrated and self consistent, with reference to the various implementational requirements of the security enforcing functions. Where lower level representations are consulted it may be possible to identify distinct components which have been traced from the different security enforcing functions; here it will be possible to confirm the binding of such components.

4.54 The expression of a binding analysis, which is required to meet those ITSEC criteria requiring consideration of 'security enforcing functions and mechanisms', will be determined in part by the granularity of the specified mechanisms. The general guidance regarding the granularity of mechanisms is applicable with regard to their role in binding, namely:

a) As a first principle mechanisms should be sought at a level of granularity equivalent to that of the security enforcing functions,

b) It is sensible to deviate from the above first principle if this is of benefit to the clarity of the logical model of the TOE provided by the set of mechanisms.

4.55 Note that expressing the mechanisms at a coarser level of granularity than that of the security enforcing functions might simplify the binding of mechanisms; however it would still be necessary to demonstrate the binding of security enforcing functions within the mechanisms, possibly involving reference to lower level components. Conversely expressing the mechanisms at a finer level of granularity than that of the security enforcing functions would, in general, complicate the binding requirements. In those cases where deviation from a strict one to one mapping of security enforcing functions and mechanisms allows a clearer logical model of the TOE to be achieved, this clearer model can be expected to give a clearer basis for binding; e.g. in terms of interactions, irrespective of the need to demonstrate binding of security enforcing functions.

Alternative Bases for Binding - E3 and above

4.56 Alternative approaches to the mechanism-based approach are acceptable provided they satisfy the requirement of the assurance level, as indicated by ITSEC Figure 4, to demonstrate binding of the security enforcing functions and mechanisms.

4.57 Typically such an approach will take as its basis the physical design components identified during Architectural and Detailed Design. The considerations of significant detail from low level representations and of binding of the security enforcing functions themselves, both of which were addressed in detail for the mechanism-based approach, are applicable in this context.

4.58 For example a binding analysis could be based on Detailed Design components, provided that there is correctness traceability evidence to justify that the set of components considered give a complete representation of the security enforcing functions and mechanisms of the TOE.

4.59 Before pursuing an alternative approach, consideration should be given to the ease with which the binding can be expressed and analysed using the proposed representation(s).

Bases for Binding - E1 and E2

4.60 At E1 and E2 ITSEC Figure 4 does not mandate consideration of the Detailed Design, in which the majority of mechanisms will be identified. In this case it would be appropriate to consider binding of the security enforcing functions specified in the security target and any supporting mechanisms identified in the Architectural Design. The Architectural Design must also be considered to take account of any detail significant to binding.

4.61 Note however that ITSEC Figure 4 does not exclude additional material being included in the basis for effectiveness analyses. At E2 therefore, where Detailed Design is required for correctness, it would still be possible to base the binding analysis on mechanisms, if this was considered to give the clearest binding model.

Application of Binding Considerations and Techniques to Bases

4.62 The binding considerations and techniques discussed in the following sections are generally applicable, within the scope of the assurance level given by ITSEC Figure 4, whichever basis is used for the binding analysis. For ease of expression the security functions, mechanisms, components (including security-irrelevant components - see below under *Security Irrelevant Considerations*) and external agents (see below under *External Interface Considerations*) which feature in the analysis are collectively referred to as 'binding elements'. However mechanisms are mentioned explicitly in contexts where this is the obvious means of expression.

Security Relevant Considerations

4.63 ITSEC 3.17-3.19 refers to 'security enforcing functions and mechanisms'. Sensibly this must be extended to cover security relevant functions and mechanisms in the following categories:

a) Mechanisms which enforce binding. Examples include:

- an address space protection mechanism, facilitating separation of different security enforcing, security relevant and security irrelevant spaces,

- a processor mode protection mechanism, restricting execution of authorised mode instructions to authorised processes,

- a process scheduling mechanism, which defends possible attack on the availability of a TOE by swapping out low priority processes and ensuring that trusted processes are given sufficient cpu time,

- an object re-use mechanism.

b) Binding elements with capability to access secure data which lies outside their immediate implementational scope, e.g. in the absence of appropriate protection mechanisms.

   Note that this effect is potentially transitive, i.e. if element 1 has accessed secure data, then element 2 might be able to access the data whilst it is held by element 1.

c) Binding elements which process secure data that must subsequently be purged.

   For example a low level, general purpose string comparison routine might be called by a high level security enforcing component to compare a user supplied password with a stored password. The string comparison routine would then need to be bound with an object re-use mechanism to ensure that the passwords did not remain in local memory. Note that, whilst low level, the string comparison routine is handling high level secure data items.

d) Binding elements which are assigned privilege.

   The assignment of privilege will allow the binding element to potentially interact with other elements in ways that would not otherwise be allowed, e.g. mechanism deactivation or access to secure data. The interactions facilitated by each privilege mode must be considered.

4.64 At assurance levels E1 and E2, for the binding to be effective, it will be sufficient to briefly state the presence of a necessary minimum of security relevant functionality required to enforce binding (see point a) above).

Security Irrelevant Considerations

4.65 Consideration of security irrelevant components can be limited to justifying the ability of the TOE to withstand any possible attack, misuse or error originating from or via such a component (e.g. a coding error causing attempted unauthorised access to secure data). The emphasis here should be on justifying the ability of the bound TCB to withstand such activity. It will typically be sufficient to include a selection of types of such components in the analysis, where the selection is representative and complete in terms of the different types of potential interactions which such components might have with the TCB.

External Interface Considerations

4.66 Binding must then be completed by justifying the ability of the TOE to withstand any possible attack, misuse or error originating from a user (e.g. an attempt by a malicious attacker to circumvent security functionality), an application (e.g. a trojan horse attack) or an external 'device' (e.g. an attempt by someone with insufficient clearance to request protectively marked data over a network link). As with security irrelevant components the emphasis should be on justifying the ability of the bound TCB to withstand such activity, considering a sufficient selection of types of such external agents.

4.67 Where lower levels of representation are used as a basis for binding, the interaction of external agents with the TOE will be via components which operate the external interfaces, e.g. a Windows MMI facility. Where such components are security irrelevant it is acceptable for them to represent the corresponding external agents within the binding analysis. In such cases the range of interaction types which the components might have with the TCB must be representative of those which might be invoked by the corresponding external agents. Where external interface components are security relevant it will be necessary for both they and representative corresponding external agents to feature in the binding analysis.

Interactions

4.68 ITSEC 3.18 requires binding to analyse ‘all potential interrelationships’ between binding elements. Theoretically this requires, for each representational level, the possible interaction(s) between every pair of binding elements to be considered. However, ‘n’ binding elements can give rise to n(n-1)/2 pairs, and for a large number of binding elements the task of analysing all possible interactions could be very onerous. Pragmatically therefore means are required to reduce the potential volume of analysis, where this can be done whilst maintaining rigour. Techniques to achieve this are discussed below under *Techniques to enable a Pragmatic Approach to Binding*. However the nature of interactions is first discussed.

4.69 The term interaction covers any form of contact between a pair of binding elements, which might in general take any of the following forms:

a) The invocation of one binding element by the other

b) The deactivation or disabling of one binding element by the other

c) Exchange of data, where any access to data maintained (i.e. derived, stored or passed) by a binding element is under the control of that element

d) Exchange of data, where any access to data maintained by a binding element is made outside the control of that element

e) Modification of one binding element by the other.

4.70 The interaction which exists between a given pair of binding elements might be multiple, e.g. involving data exchange after invocation or involving more than one data exchange. A sequence of related interactions can usually be considered sensibly as a single, compound interaction. Where two interaction sequences are unrelated however, it is clearest to consider them as distinct interactions.

4.71 In addition to considering the effect of direct interactions between binding elements, it is also necessary to consider the consequent indirect interactions, i.e. if binding element 1 interacts directly with binding element 2, and if binding element 2 interacts directly with binding element 3, then element 1 interacts indirectly with element 3. In general, there is then scope for further indirection of interactions. For completeness, the full scope of the effect of indirect interactions, the ‘closure’ of the ‘transitivity’, should be considered.

4.72 Interactions involve shared access to trusted resources, where resources can be generalised to take the forms of data items, binding element code, scheduling directives and privileges. Indirect interactions are thus seen to involve indirect sharing of resources, e.g. the passing of secure data via one or more binding elements.

4.73 All potential interactions must be considered, including any which are not required for correctness of the implementation, e.g. involving, at E3, an unused operating system call.

4.74 For the purposes of binding analysis it is helpful to recognise that interactions can be categorised as follows:

a) Mandatory

A mandatory interaction is one which is required to enforce binding. As such it will involve two trusted binding elements.

For example, where it is required to audit an attempted access to an object which is subject to access rights, there is a mandatory interaction between access control and audit, involving the passing of data regarding the attempted access.

b) Forbidden

A forbidden interaction is one which would invalidate the binding, and as such must be prevented. It will typically involve a trusted and an untrusted element. Indirect attacks originating from untrusted agents constitute Forbidden interactions.

For example, a security irrelevant component would be forbidden from reading stored passwords maintained by an authentication mechanism.

c) Permissible

A permissible interaction is one which is neither required to enforce the binding or such as to invalidate it. It will involve either a trusted and an untrusted element or a non-security related interchange between two trusted elements.

For example the interaction between a user and the Identification & Authentication mechanism via the log on MMI is permissible. Note that this interaction is not mandatory as the TOE would remain bound if the user were to decline the opportunity to attempt to log on.

4.75 Having determined and categorised the interactions it is necessary to show that Mandatory interactions are enforced and Forbidden reactions are prevented. In the case of permissible interactions it is sufficient to merely categorise them as such.

4.76 The above categorisation has been used successfully in binding analyses. Whilst not mandated, it is recommended. The following discussion assumes that such a categorisation is used.

Techniques to enable a Pragmatic Approach to Binding

4.77 As indicated above under *Interactions*, the task of analysing all possible interactions could be very onerous; techniques are thus required to reduce the volume of analysis, where this can be done whilst maintaining rigour. A number of such techniques are suggested in this section. The section is subdivided into two, discussing first techniques to be employed within the binding analysis, then discussing techniques integral to the design of the TOE itself.

Techniques to enable a Pragmatic Approach to Binding - (1) Analysis

4.78 A primary technique, to reduce the number of potential interactions to be considered whilst maintaining rigour, is that of identifying interaction types, each of which is representative, in all significant respects, of a number of interactions. Typically such interaction types are associated with mechanisms which provide a general service to the whole TCB, and thereby provide countermeasures against particular modes of indirect attack. If a generic rationale can be provided to justify that this service is enforced across the TCB, then binding of the mechanism with all appropriate binding elements is confirmed.

4.79 For example, consider an address space protection mechanism, which protects both instruction and data spaces. Mandatory interactions exist between this mechanism and trusted binding elements, the protection mechanism supporting the binding element in each case by protecting the binding element's executable code from corruption. Upholding of each mandatory interaction will in turn exclude the Forbidden interactions, involving corruption of trusted binding element code, between each such trusted binding element and the representative set of security irrelevant components and external agents. Additionally parallel Mandatory and Forbidden interactions exist in respect of secure data controlled by the trusted binding elements. Sensibly the protection mechanism can be expected to be implemented in such a way that it is applied to all trusted binding elements. It is sufficient for the binding analysis to include a rationale, confirming that this is so, instead of repeated consideration of the Mandatory interactions which the mechanism upholds and the Forbidden interactions which it excludes.

4.80 Other mechanisms which can be expected to be associated with representative interaction types include:

a) Processor mode protection mechanisms

b) Process scheduling mechanisms

c) Object re-use mechanisms

d) Access control mechanisms.

4.81 It is unnecessary to over complicate the binding analysis by needlessly composing nonsensical potential interactions between trusted binding elements, e.g. the passing of a copy of audit data from an audit logging mechanism to an authentication checking mechanism. Such quirks are unlikely. Any which do exist should manifest themselves either during the consideration of high level binding elements to identify associated Mandatory and Permissible interactions or as a traceability issue: their effect on binding can then be considered.

4.82 At assurance levels E4 and above it may occasionally be necessary to refer to source code when considering binding, e.g. if a basic component implements more than one security enforcing function. Two techniques merit consideration here:

a) Module test results may be used to justify the effectiveness of Mandatory interactions, provided that the testing is adequate to demonstrate such binding.

b) Adherence to good coding practice may be used to help justify that Forbidden interactions do not exist, e.g. use of structured programming techniques, provision of clearly defined interfaces between code sections, and avoidance of GOTO statements and pointers may be used to justify the effective separation of code sections.

4.83 At assurance level E6 consideration of object code may be restricted to those areas where inconsistencies between source code and object code are suspected.

4.84 It may be possible, on a case by case basis, for the construction vulnerability analysis to argue that binding vulnerabilities are not exploitable in practice. For example, at a low level of representation where an MMI component has been classified as security irrelevant within a system TOE, the possibility of secure data passed by the MMI component being exploited by a trojan horse attack could be identified as a binding vulnerability. However it might be possible for the construction vulnerability analysis to deem this to be of no significance, by reference to the secure development of the system integrator and the fact that the particular use of the MMI component was unknown at those times when it could have been effectively corrupted. Note that in this case it would still be possible for the binding analysis to demonstrate that the MMI component was subject to other binding considerations, e.g. the control of an object re-use mechanism.

Techniques to enable a Pragmatic Approach to Binding - (2) Design

4.85 It should be appreciated that a good security-oriented design can restrict the number of interactions and thereby simplify binding. A number of design techniques, available to the developer, are of use in this respect.

4.86 A 'reference monitor' is defined, within this context, to be a binding element which performs and controls a particular type of interaction. For example, a reference monitor may control:

a) access to secure data,

b) process activation and deactivation.

4.87 In such cases it will be sufficient for the binding analysis to include:

a) a rationale, justifying that the interaction is policed by the reference monitor in such a way as to exclude undesired interactions,

b) consideration of desired (Mandatory and Permissible) interactions with the reference monitor.

4.88 Reference monitors may be layered, with a high level reference monitor dependent on a lower level reference monitor. For example, a file I/O reference monitor may be dependent on a disk I/O reference monitor. Typically lower level reference monitors are more permissive than higher level reference monitors; it is thus necessary for clients of the high level reference monitor to be prevented from interacting with the lower level reference monitor to ensure that the high level controls are not circumvented and traceability is preserved.

4.89 A 'gate' is defined, within this context, as that part of a binding element which can interact with other elements. Where a binding element has only a necessary minimum of well defined gates the number of associated interactions to be considered in the binding analysis will also be minimised. Note that gates are traceable across different levels of correctness representation.

4.90 The use of gates to minimise the number of direct interactions is of consequent benefit in reducing the number of indirect interactions.

4.91 Well defined gates can also be used to reduce the potential effect of indirect interactions. In general, if binding element 1 has read access to trusted resource A and modify access to trusted resource B, and if binding element 2 has read access to trusted resource B, then element 2 has indirect read access to resource A. However if the gate of element 1, which specifies the modify access to resource B, is well defined so as to exclude the passing of any content of resource A, and if correctness analysis for element 1 demonstrates that no inference from the reading of resource A is made when modifying resource B, then this aspect of transitivity need be considered no further, with consequent simplification of the overall transitivity closure.

4.92 Where there is potential for indirect interaction between two binding elements, resulting from the direct interaction of a third element with first one and then the second element in different modes, e.g. privileged and non-privileged modes, this will usually have been countered by the use of an object re-use capability to 'flush' the third element between modes. In such cases the possibility of the indirect interaction, which would often be Forbidden, can be excluded.

Notation

4.93 At levels of assurance E1 to E3, where the binding analysis is based on correctness deliverables using an informal specification style, the binding analysis may be presented in an informal style.

4.94 At levels of assurance E4 to E6 notations which allow the binding to be expressed more rigorously may be used. The options available include:

a) Flow diagrams (e.g. Yourdon, SSADM)

   These depict binding elements as symbols and interactions as interconnecting lines. They have the advantage of making it easier to visualise interactions, especially indirect interactions. Pragmatically, the diagrams may be clarified by omitting Forbidden interactions and instantiations of interactions of a representative type; however there remains a requirement for the binding analysis to adequately consider all necessary interactions.

b) Matrices (e.g. Kemmerer's shared resource matrix)

   These typically use columns to represent the binding elements, rows to represent trusted resources, and cells to represent the direct access rights which binding elements have to resources. The sharing of trusted resources thus express the interactions between binding elements. The matrix can be further populated in respect of indirect interactions, and provides a tool for systematic analysis of the transitivity closure.

Covert Channel Analysis

4.95 Wherever covert channels are a threat, e.g. if the TOE has confidentiality objectives and a MAC policy, then they must be assessed. Note that this applies irrespective of whether covert channels are explicitly mentioned by the security target.

4.96 Strictly, the identification of covert channels is a binding issue and consideration of their exploitability is a construction vulnerability issue. In practice, it might be easiest for all covert channel issues to be considered together, with references made from the binding and construction vulnerability deliverables as appropriate.

4.97 Covert channels take the form of storage channels (which use data stores to covertly transfer information) and timing channels (which use event timings to covertly transfer information). Each of these represents a type of interaction.

4.98 For example, where a flag is used to control access to an object by more than one subject, it is possible for one subject to pass information to another by setting or clearing the flag. With synchronisation a complete bit stream could be passed, thereby covertly exploiting the storage channel offered by the flag.

4.99 In assessing the potential exploitability of a covert channel, the significance of its bandwidth should be assessed on a case by case basis. For some TOEs a very small bandwidth may represent a serious problem.

4.100 At assurance levels E1 to E3, where covert channels are a possible threat, there must be some evidence that covert channels have been considered. This will typically take the form of an informal argument for why covert channels are not exploitable or why they do not represent a significant risk.

4.101 At E4 and E5, where covert channels are a possible threat, the following is required:

a) A description of the method used to search for covert channels, and evidence that a thorough search has been performed,

b) A description of all covert channels found,

c) An estimate of the bandwidth of each covert channel,

d) A discussion of the exploitability of each covert channel.

4.102 At E6, where covert channels are a possible threat, a more rigorous approach is required. For example, Kemmerer's shared resource matrix method may be used to search for covert channels. Covert channel bandwidths should be estimated accurately, and if possible should be supported by actual measurements. Thorough and complete arguments about their exploitability must be given.

Strength of Mechanisms Analysis

Description

4.103 A security mechanism is defined in the ITSEC as the logic or algorithm that implements a particular security enforcing or security relevant function in hardware and software.

4.104 The strength of mechanisms (SoM) analysis is concerned with demonstrating the ability of a security enforcing mechanism to withstand a direct attack which attempts to exploit any deficiencies in the underlying algorithms, principles, or properties of the mechanism.

4.105 The SoM analysis should list all mechanisms that have been identified as critical within the TOE, and should include references to the underlying algorithms, principles, and properties of those mechanisms. It should show that all critical mechanisms satisfy the claimed minimum strength of mechanisms rating. (See definition of *Critical Mechanisms* later in this section.)

4.106 The evaluators will perform penetration testing during the evaluation to confirm or disprove the claimed strength of mechanisms.

Strength Ratings

4.107 The strength of a critical mechanism is represented by one of three ratings: *basic*, *medium*, or *high*. These are defined as follows:

a) basic: the mechanism provides protection against random accidental subversion, although it may be capable of being defeated by knowledgeable attackers

b) medium: the mechanism provides protection against attackers with limited opportunities or resources

c) high: the mechanism can only be defeated by attackers possessing a high level of expertise, opportunity and resources, successful attack being judged to be beyond normal practicability.

4.108 These definitions do not provide a direct, explicit or testable means of assessment during evaluation. The rating of the strength of mechanisms is based upon a consideration of the aspects of expertise, opportunity (in terms of time and available equipment), and resources (including collusion with others). These indicate the distinction between the amount of effort needed to discover a vulnerability and the amount of effort necessary to exploit it.

4.109 When assessing the strength of a mechanism, the developer should consider all reasonable combinations of the above. The ITSEM refines these terms further, providing tables to help calculate the appropriate strength. This information is reproduced below.

4.110 **Expertise**, the knowledge required by the attacker, is categorised as:

a) layman: no particular expertise,

b) proficient: familiar with the internal structure of the TOE,

c) expert: familiar with the underlying principles and algorithms of the TOE.

4.111 **Resources**, the resources necessary for a successful attack, are divided into the equipment required, and the time taken, to perform an attack. The equipment necessary can be one of:

a) unaided: no special equipment,

b) domestic: equipment readily available within the TOE environment, part of the TOE itself, or available to the public,

c) specialist: specialised purpose-built equipment for the attack.

4.112 The time taken can be one of:

a) minutes: less than ten minutes,

b) days: less than one month,

c) months: more than one month.

4.113 Opportunity covers factors which are generally outside an attacker's control, including the possibilities for collusion, chance circumstances, and the likelihood of detection. Only collusion is categorised in the ITSEM, but the other factors must also be considered when justifying statements made about the strength of mechanisms. Collusion can be one of:

a) alone: no collusion,

b) user: the attacker requires help from an untrusted user,

c) administrator: collusion with a highly trusted user or administrator is required.

4.114 The tables provided in the ITSEM are reproduced below. The strength rating for a mechanism can be calculated by adding together the numbers found using the first two tables, and comparing the result against the third table.

Defensive Strength	Collusion			
Time		alone	with a user	with an administrator

Defensive Strength	Collusion			
	within minutes	0	12	24
	within days	5	12	24
	months/years	16	16	24

Required Attacking Strength	Equipment			
Expertise		unaided	using domestic equipment	using specialist equipment
	layman	1	n/a	n/a
	proficient	4	4	n/a
	expert	6	8	12

Result	Strength of Mechanism
1	none
2 - 12	basic
13 - 24	medium
25 - 36	high

Figure 5 Calculating the Strength of Mechanisms

4.115 Thus, a mechanism which is defeated by a proficient attacker alone and unaided within minutes (0 + 4) would be classed as basic. A mechanism which is defeated in months by an expert using specialist equipment and helped by a user (16 + 12) would be rated high.

4.116 Note, however, that the tables provide only guidance and their applicability to a particular mechanism and environment must be considered in context by the author of the SoM analysis.

4.117 The claimed strength of the weakest critical mechanism in the TOE is used in the security target to specify the overall minimum strength of mechanisms rating for the TOE. The SoM analysis should identify the critical mechanisms and calculate their strength.

Considerations

4.118 Because operational factors, such as the possibility of collusion with a user or with an administrator, are included in the SoM rating scheme, there is an overlap between the SoM and operational vulnerability analyses. It is the responsibility of the author of the effectiveness analyses to justify where such aspects are best considered. If a vulnerability analysis identifies a critical mechanism as the major countermeasure to a vulnerability, then the strength of this mechanism must be confirmed in the SoM analysis.

The Context of Strength of Mechanisms

4.119 The construction correctness analyses check the correct realisation of individual security enforcing functions from their specification in the security target to their implementation in source code or hardware. The construction effectiveness analyses first check the adequacy of the security enforcing functions to counter the threats, and the ability of the combination of the security enforcing functions and their realisations to provide a secure, integrated whole. This still leaves the possibility that a weakness might exist in the form of a mechanism which, because of the nature of its conception, can be overcome by direct attack. The strength of mechanisms analysis is required to identify such mechanisms and assess their ability to withstand direct attack. Note that the strength of mechanisms analysis thus assesses the conception of mechanisms conceived to provide particular security functionality, rather than considering vulnerabilities which may result from the inadequacy, poor combination or incorrect realisation of security enforcing functions.

4.120 The claim for a minimum strength of mechanisms is seen to complement the specified assurance level, by making a claim in respect of those mechanisms which, by nature of their essential conception, each contain a weakness which leaves them vulnerable to direct attack.

4.121 The minimum strength of mechanisms claim also provides a metric, consistent with the threats specified in the security target, which may be used to determine whether vulnerabilities in the TOE are exploitable in practice.

4.122 The strength of mechanisms analysis applies to security enforcing mechanisms. The following discussion therefore first confines its attention to security enforcing mechanisms. However the role of security relevant mechanisms in the context of the minimum strength of mechanisms claim is then considered.

Direct Attack

4.123 In general, an attack is an attempt to violate the TOE's security objectives by exploiting a weakness of one of the following types:

a) a flaw in traceability and implementation (i.e. where the TOE is not correct),

b) the inability of countermeasure(s) to adequately counter a threat (i.e. where the TOE is not suitable),

c) failure of the TOE's security enforcing functions and mechanisms to provide an integrated and effective whole (i.e. where the TOE is not effectively bound),

d) a mechanism which, because of the essential nature of its conception, possesses a residual weakness in its underlying algorithm, principles or properties,

e) insecure operation which is not easily detectable by the authorised user (i.e. where the TOE's ease of use is not effective).

4.124 Neither the ITSEC nor the ITSEM define direct attack. However both refer to direct attack in the context only of strength of mechanisms analysis. Therefore by inference, a direct attack is understood to be an attempt to violate the TOE's security objectives by exploiting a weakness in the underlying algorithm, principles or properties of a particular mechanism.

4.125 Direct attack typically involves manipulation of inputs to and/or outputs from the mechanism which are within its specification.

Critical Mechanisms

4.126 A *critical mechanism* is defined as a *security enforcing mechanism within the TOE which is susceptible to direct attack.*

4.127 Note that this definition constitutes a revision of that given by the ITSEC. The revised definition does however allow an interpretation of the requirements for strength of mechanisms analysis which is consistent with the purpose of the ITSEC for strength of mechanisms analysis (which is the only ITSEC aspect in which critical mechanisms feature). It is considered clearer and preferable to the definition given by the ITSEC, which has proved to be problematic in practice.

Type A Mechanisms

4.128 ITSEM 6.C.4 defines a *type A mechanism* as a *security mechanism with a potential vulnerability (weakness) in its algorithm, principles or properties, whereby the mechanism can be overcome by the use of sufficient resources, expertise and opportunity in the form of a direct attack.*

4.129 ITSEM 6.C.7 defines a contrasting *type B mechanism* as a *security mechanism which, if perfectly conceived and implemented, will have no weaknesses.*

4.130 These definitions are mutually exclusive, with the distinction between type A and type B mechanisms being seen to pertain to the different natures of the mechanism types. Whereas a type B mechanism can be conceived and implemented so as to have no weakness, the essential concept of a type A mechanism is such that a residual weakness will exist. For a type A mechanism there remains a requirement for the detailed conception to introduce no additional weakness and for the conception to be correctly implemented.

4.131 Type A mechanisms are seen to be those which typically involve the use of a 'secret', such as a password or cryptographic key.

4.132 A type A mechanism is equivalent to a critical mechanism where it is security enforcing. The type A definition and the distinction between types A and B thus amplifies the critical mechanism definition.

Mechanism Granularity

4.133 For the purpose of the strength of mechanisms analysis, the granularity of critical mechanism which should be identified is *that which, if subject to a successful direct attack, would permit the violation of a security enforcing function, provided that the subjection of a submechanism to the same direct attack would not lead to a violation of the same security enforcing function.*

4.134 For an amplified understanding of the above definition the following points should be considered:

a) The coherence of a mechanism might be such that its submechanisms are not directly attackable. In such a case it would not be sensible to assess a granularity lower than that of the mechanism, as this would constitute disregard for the binding of the submechanisms.

b) Defining a mechanism at too coarse a level of granularity would serve only to complicate a strength of mechanisms analysis. The mechanism would still require analysis to determine which submechanisms could be subject to which attacks, and to factor out a clear understanding of directly attackable submechanisms to assist their strength assessment.

4.135 Where use of a certain security enforcing mechanism is prescribed by the security target it is to be expected that this is a coherent mechanism which implements security enforcing function(s) specified elsewhere in the security target. As such the granularity of the prescribed mechanism can be expected to be that which can be identified by consideration of the potential violation of the appropriate security enforcing function(s), as discussed above. Where this is not the case then the specification of the security enforcing function(s) and prescribed mechanism should be reconsidered.

4.136 Non-critical mechanisms should be identified at a level of granularity giving a clear abstraction which can be used to support the justification that they are indeed non-critical.

Cryptographic Mechanisms

4.137 ITSEC 3.23 states that in the case of cryptographic mechanisms, the appropriate national body shall provide a statement of confirmation of the minimum strength. In the UK the appropriate national body is CESG; liaison with CESG should initially be via the certifier.

4.138 CESG considers the following mechanisms to come within the definition of cryptographic mechanisms (note that this list may be expanded in future):

a)	confidentiality mechanisms	- encryption - decryption
b)	integrity mechanisms	- hashing
c)	authentication mechanisms	- password generation - password encryption

- challenge and response
- digital signature

d) non repudiation mechanisms

e) random number generators

4.139 All the above mechanisms are implemented by means of an algorithm which may be publicly known, proprietary or for HMG use only. Certain mechanisms (e.g. digital signature) may be implemented by an algorithm which uses public key cryptography.

4.140 Whilst cyclic redundancy checksums offer protection against unintentional data errors, CESG considers them to have little strength, and does not consider them to be cryptographic mechanisms. The sponsor is advised that use of such mechanisms should be constrained to protection against unintentional data errors.

CESG Position on Cryptographic Algorithms

4.141 CESG, as the national body responsible for providing advice on cryptography for use by HMG, prescribes the use of a limited number of algorithms. These algorithms may only be obtained from CESG, and products and systems containing them may only be supplied to users with CESG permission. CESG advises users directly on the appropriateness of algorithms for their applications. The strength of algorithm is described by level - algorithms may be *baseline, enhanced* or *high grade*. CESG has never seen any point in stating a strength for these mechanisms as it would only serve to cause confusion.

4.142 Where CESG rates a proprietary mechanism, it will only confirm that the mechanism has a strength commensurate with the assurance level of the TOE. The following table applies:

Assurance Level	Appropriate Strength of Mechanism
E1	Basic
E2	Basic or Medium
E3	Medium or High
E4, E5, E6	High

Figure 6 Assurance Levels and Commensurate SoM Ratings

4.143 CESG has a long-standing policy of not commenting on the strength of mechanism of publicly known algorithms. Their strengths and weaknesses are well documented. It is also CESG policy not to comment on public key cryptography and random number generators.

4.144 All mechanisms other than those for password generation will be treated as follows:

a) For HMG use, CESG will always evaluate the strength and implementation of the algorithms and its work may exceed what is normally done at a specific ITSEC assurance level. Associated functionality will often require evaluation within CESG, who will provide support to the CLEF evaluation in the form of directives, i.e. questions that need answering.

b) For non-HMG use, all proprietary algorithms for non-public key cryptography, with the exception of random number generators, will be evaluated by CESG, whose work will include at least a crypto check. The CLEF will not need access to algorithmic details, and its work will be restricted to more peripheral matters, which may include questions asked by CESG.

c) For non-HMG use, random number generators will not have their strength of mechanism rated. The CLEF will, however, use supplied data on the algorithms to confirm their correct implementation.

d) Algorithms for non-HMG use, publicly known algorithms and public key cryptography algorithms will not have their strength of mechanism rated. The CLEF will however use published data on the algorithms to confirm their correct implementation.

4.145 Password generation mechanisms will be treated as follows:

a) For HMG use, CESG will always evaluate the strength and implementation of the algorithms. Associated functionality will often require evaluation within CESG, who will provide support to the CLEF evaluation in the form of directives, i.e. questions that need answering.

b) Algorithms for non-HMG use, proprietary and publicly known algorithms will be evaluated by the CLEF. CESG is not involved in these cases. The CLEF evaluation will involve an analysis of password space, as outlined below in *Example 3 - Analysis of Password Space*.

4.146 If the TOE contains no critical mechanisms or if critical mechanisms are either to be evaluated by CESG or not to have their strength rated, then this must be stated in the security target. Appropriate wording should be selected from the following table, which also indicates the equivalent wording which will be included in the Certification Report:

Case	Security Target	Certification Report
HMG use - password generation	The algorithm must be approved by CESG for *specified use*	The algorithm is approved by CESG for *specified use*
HMG use - other than password generation	The algorithm must be approved by CESG for the protection of *specified type of* information.	The algorithm is approved by CESG for the protection of *specified type of* information.
non HMG - proprietary - other than password generation, public key cryptography or random number generation	The *named* algorithm must be rated by CESG as having a strength commensurate with the assurance level. (Note that this wording is also appropriate, in addition to the strength claim, for the SoM Analysis)	The *named* algorithm is rated by CESG as having a strength commensurate with the assurance level.
non-HMG - publicly known (not password generation) - public key cryptography (not password generation) - random number generation	The *named* algorithm *is publicly known/uses public key cryptography/ is a random number generator* and as such no comment will be made on its appropriateness or strength	The *named* algorithm *is publicly known/uses public key cryptography/ is a random number generator* and as such no comment has been made on its appropriateness or strength.
No Critical Mechanisms	All mechanisms are non-critical.	All mechanisms have been confirmed to be non-critical.

Figure 7 Wording for Special Critical Mechanism Cases

Minimum Strength of Mechanisms

4.147 The minimum strength of mechanisms claim must be commensurate with the assurance level of the TOE. See Figure 6 for details.

4.148 If the TOE contains critical mechanisms which are either proprietary to be evaluated by CESG or are to be evaluated by the CLEF then the minimum strength of mechanisms claim must be no higher than the minimum strength of such mechanisms. Where proprietary mechanisms are to be evaluated by CESG and the assurance level is either E2 or E3, then the minimum strength of mechanisms claim must be no higher than the lower of the two strengths which are commensurate with the assurance level.

4.149 In all cases, including those where the TOE contains no critical mechanisms, or where all critical mechanisms are either to be evaluated by CESG or not to have their strength rated, then the minimum strength of mechanisms claim is available as a metric which may be used to determine whether vulnerabilities in the TOE generally are exploitable in practice.

a) This is relevant to the construction and operational vulnerability assessments, as described below under *Relationship to Vulnerability Assessment*.

b) It is also relevant within the context of the penetration testing performed by the evaluators. Note that this may cover ad hoc tests, in accordance with ITSEM 4.5.103, in addition to tests concerned with potential vulnerabilities identified earlier in the evaluation.

Strength of Mechanisms Analysis

4.150 The strength of mechanisms analysis must:

a) Identify any critical mechanisms (at the appropriate level of granularity) within the TOE.

b) Include, or reference, the underlying algorithms, properties and principles of each critical mechanism (except in the case of mechanisms intended for HMG use only).

c) Show that each critical mechanism satisfies the claimed minimum strength of mechanisms rating stated in the security target (except in the cases of mechanisms either intended for HMG use only or not to have their strength rated).

d) Identify all other security enforcing mechanisms within the TOE and justify, to the rigour required for the level of assurance, their categorisation as non-critical. Note that this justification will often be straightforward, and for certain TOEs it might be pragmatic for the identification and categorisation of non-critical security enforcing mechanisms to be addressed in other deliverables (e.g. in the detailed design documentation). In the latter case reference to the justification must be provided in the SoM analysis.

4.151 The sponsor must provide sufficient evidence to support a strength of mechanism assessment. At E1 such evidence may require more documentation than that explicitly required by the ITSEC; for instance, a description of the detailed design may be necessary in order to support the assessment.

Relationship to Binding Analysis

4.152 The effect of indirect attacks, including those involving the deactivation, disabling, corruption of and tampering with a mechanism, is outside the scope of the strength of mechanisms analysis. The protection offered by other mechanisms is covered under binding analysis.

4.153 For the minimum strength of mechanisms claim to be confirmed, it is a prerequisite that the TOE is effectively bound. The strength of mechanisms analysis should therefore reference the binding analysis, in order to demonstrate the categorisation of critical and non-critical mechanisms by confirming which are directly attackable.

Relationship to Vulnerability Assessment

4.154 It must be understood that the minimum strength of mechanisms rating makes an implicit assertion about the TOE's environment, regarding the nature of attacks, in terms of the levels of expertise, opportunity and resources, which constitute threats. For example, if the minimum strength of mechanisms claim is 'basic', then it follows from ITSEC 3.6 that knowledgable attackers are not considered a threat in the TOE environment.

4.155 This implied assertion must be consistent with the threats specified in the security target, i.e. the threats must not require a higher minimum strength of mechanisms countermeasure than that actually claimed.

4.156 Determination of whether a vulnerability is exploitable in the TOE's environment may then involve consideration of the levels of expertise, opportunity and resources required for its exploitation. Where these are less than those associated with the minimum strength of mechanisms claim then the vulnerability will be exploitable.

4.157 Where the minimum strength of mechanisms claim is used to justify that a vulnerability is not exploitable in practice, then it must be clearly specified why the minimum strength of mechanisms metric is appropriate to the justification.

Security Relevant Mechanisms

4.158 The strength of mechanisms analysis is concerned with security enforcing mechanisms. However the role of security relevant mechanisms within the TOE must be addressed by the appropriate evaluation work packages. Any security relevant type A mechanisms should be considered in context, on a case by case basis, and the Certification Body should be contacted to ensure that the proposed approach is acceptable. In general it is to be expected that security relevant type A mechanisms will be addressed by the construction and/or operational vulnerability assessment(s). Consider the following examples:

a) A type A mechanism employed to guarantee the authenticity of the delivered TOE would be addressed by the operational vulnerability assessment.

b) Suppose that DAC security enforcing functions are met by an ACL mechanism. Suppose further that a token passing DAC mechanism (e.g. password protection of a database table column), which does not meet the DAC security enforcing functions, exists in addition to the ACL mechanism. The token passing mechanism is then not a parallel security enforcing DAC mechanism, but is security relevant in that it could potentially be exploited to circumvent the security enforcing ACL mechanism. It would thus be addressed by the construction vulnerability assessment.

4.159 Determination of the sensible granularity of a security relevant mechanism will involve consideration of the granularity which can coherently support the security of the TOE and that which can be subjected to direct attack.

4.160 Where a security relevant type A mechanisms falls within the scope of cryptographic mechanisms, as outlined above under *Cryptographic Mechanisms*, then the same considerations apply.

Example 1 - Password with Retry Limitation

4.161 This is the example, given in ITSEM 6.C.9 to 6.C.11, of a mechanism which comprises password and retry limitation submechanisms. The role of the retry limitation is seen to be that of limiting the scope for direct attack on the primary password submechanism. The mechanism to be assessed is thus seen to be the combined password with retry limitation mechanism; this combined mechanism is critical.

4.162 Note that the password submechanism is of type A, and would also be coherent and critical in the absence of the retry limitation submechanism. Indeed it could be argued that the password submechanism is directly attackable, and therefore critical, in the presence of the retry limitation submechanism. However it would then have to be assessed in the context of the retry limitation submechanism, so it is sufficient and clearer to identify the combined mechanism as critical.

4.163 The retry limitation submechanism is only meaningful when combined with the password submechanism. To support the strength assessment of the combined mechanism the binding analysis must confirm that the retry limitation submechanism cannot be bypassed.

Example 2 - Password File

4.164 Consider the protection of a password file by MAC, DAC and encryption of the passwords.

4.165 Suppose first that a security enforcing function claims that plain text passwords cannot be viewed by a malicious user. Suppose further that the binding of the TOE is effective and such that the MAC, DAC and encryption constitute three submechanisms which would have to be overcome in turn. The mechanism to be assessed is thus seen to be the combined MAC, DAC and encryption mechanism; this combined mechanism is non-critical.

4.166 Note that the first submechanism, MAC, is of type B and this is sufficient to ensure that the assessed mechanism is non-critical. This holds equally in respect of DAC. In this context the encryption submechanism, which is of type A but not directly attackable, will not make the assessed mechanism critical for an effectively bound, correct TOE.

4.167 Suppose also that another security enforcing function claims that plain text passwords cannot be viewed by the TOE's administrator. Assuming that the administrator is able to override both MAC and DAC, this claim is met by the encryption alone. In respect of this claim the encryption mechanism is the mechanism to be assessed, and is critical.

Example 3 - Analysis of Password Space

4.168 Where a type A mechanism depends on a 'secret' of some description (e.g. a cryptographic key, or a password), it is important that the statistical measurement of its strength is based on the size of the effective space from which values of the 'secret' are chosen. In particular, in cases where users are allowed to select their own 'secret' values (as may be the case with passwords), the effective space may be much smaller than the available space, because users are likely to choose values which are in some way memorable. In order to determine the strength of such a mechanism, the accepted practice is to assume the worst case, for example that users will choose passwords which are simple to remember (and to guess) if they are allowed to do so.

4.169 Thus, a password mechanism which allows users to select a 4-digit number as a password (on the principle that a PIN number is sufficient protection for a bank account and therefore for a user account) has an effective space of just 10000 possible passwords. If the users are allowed to choose four-letter dictionary words, the effective space is about 5000 passwords. This is easily breakable within minutes by an expert using specialist equipment: at 100 attempts per second to log on, the 4-digit 'PIN number' is broken in less than two minutes, and the 4-letter dictionary word in less than one minute. In either case, the mechanism therefore achieves only a strength of basic.

4.170 By contrast, an algorithm in which the system generates 'CLEARVIEW' passwords of the form CVCCVCCVC, where C denotes one of 20 consonants and V denotes one of 6 vowels (A, E, I, O, U, Y) has an effective password space equal to its apparent password space, equal to $20^6 \times 6^3$ = 13824 million possible passwords. Against an automated log on attack operating at a rate of 2000 attempts per second, the attack would need to last for 80 days or about two-and-a-half months, to be certain of success. If the attack can operate only at 100 attempts per second, it would need to continue for more than four years to be certain of success. Both of these are long enough to achieve a strength rating of high.

4.171 Of course, secondary mechanisms can assist in frustrating such an attack. One example would be a mechanism which raises an alarm after a small number of consecutive failed attempts to log on, thereby increasing the probability that the attack will be detected while still in progress. Another example would be a mechanism which forces a delay between log on attempts, and thereby limits the rate at which an attack can be made, without causing serious inconvenience to the legitimate user. Both of these secondary mechanisms are of type B, but must be considered in the final analysis as they have an impact on the strength of a type A mechanism.

4.172 The rightmost four columns in the table below represent four password schemes: a 4-letter dictionary-word password, a 4-digit 'PIN number', a password taken from a full English dictionary, and a CLEARVIEW password. Each of these schemes has an effective password space (a number of possible passwords), given in the second row of the table. The fourth and fifth rows of the table represent two possible automated attacks running at 100 attempts per second and at 2000 attempts per second, and the third row shows the effect of forcing a short (half-second) delay between attempts.

4.173 For each attack and each password scheme, the table gives the time required for an attack to be certain of searching out a password, together with the resulting estimate of the strength of the mechanism. These time estimates are obtained by dividing the size of the password space by the number of attempts possible per second (to obtain the maximum time for which the mechanism can resist the attack), and then dividing by two to obtain the average time for which the mechanism will resist the attack.

	4-letter dictionary words	4-digit 'PIN'	All Dictionary Words	CLEARVIEW
Effective password space	5000	10000	250000	13824 million
Average time to resist attack where delay limits attack rate to two attempts per second	21 minutes (medium)	42 minutes (medium)	17 hours (medium)	110 years (high)
Average time to resist automated attack at 100 attempts per second	25 seconds (basic)	50 seconds (basic)	21 minutes (medium)	2 years (high)
Average time to resist automated attack at 2000 attempts per second	1.3 seconds (basic)	2.5 seconds (basic)	63 seconds (basic)	40 days (high)

Figure 8 Example Analysis of Password Space

Use of Cryptographic Algorithms

4.174 If a developer is considering designing or marketing a product which incorporates an algorithm to protect HMG information, they should initially contact the Certification Body Secretariat at the address shown at the front of this document.

Vulnerability Analysis

Description

4.175 Before and during the other aspects of evaluation of the TOE, various vulnerabilities in the construction of the TOE (such as ways of deactivating, bypassing, corrupting, or circumventing security enforcing functions and mechanisms) will have been identified by the developer.

4.176 The Certification Body maintains a database of known vulnerabilities. The sponsor or developer must request relevant information from the database for their TOE by application to the Certification Body secretariat at the address given at the front of this Guide. Such requests must be submitted in a standard format, an example of which is shown at the end of this chapter.

4.177 For the construction vulnerability aspect of effectiveness these known vulnerabilities must be assessed to determine whether they could in practice compromise the security of the TOE as specified in the security target. The list of vulnerabilities provided by the sponsor must identify all known vulnerabilities (including those provided by the Certification Body) and provide an assessment of their potential impact, identifying the measures proposed to counter their effects. The analysis of the potential impact of each vulnerability must show that the vulnerability cannot be exploited in the intended environment for the TOE either because the vulnerability is covered by other security mechanisms, or because the vulnerability is irrelevant to the security target or is adequately countered by other procedural or physical security measures as documented in the security requirements.

4.178 The operational vulnerability analysis is very much like the construction vulnerability analysis except that it is applied to known operational vulnerabilities. These vulnerabilities will normally have been identified by the developer during development and testing. The developer must identify each known vulnerability, provide an analysis of its potential impact, and identify the measures proposed to counter its effect.

4.179 The evaluators will check the correctness of each analysis and perform independent vulnerability analyses, including penetration testing to confirm or disprove whether the vulnerabilities are exploitable. All combinations of vulnerabilities should have been addressed and no undocumented or unreasonable assumptions about the intended environment should have been made. The evaluators will have a copy of the information provided by the Certification Body from their vulnerability database. Note that this information may be rechecked by another request from the evaluators to the Certification Body just prior to performing penetration testing.

Approach

4.180 The construction vulnerability analysis could be divided into two sections, the first identifying all known vulnerabilities in the construction, and the second describing whether each vulnerability can be exploited, and if so how the exploitable vulnerabilities are countered. This can either be done for the document as a whole, identifying all the vulnerabilities in one place before describing their exploitability and the countermeasures needed, or it can be done for each vulnerability in turn, providing identification following by analysis for each vulnerability one after another.

4.181 Construction vulnerabilities can be separated into vulnerabilities in the effectiveness of the TOE, in particular binding errors, and known weaknesses in the design and implementation, discovered during the development, testing and review of the TOE, which may cause it to function insecurely.

4.182 The operational vulnerability analysis could also be divided into two sections, the first identifying all known vulnerabilities in the operation of the TOE, and the second describing whether each operational vulnerability can be exploited, and how exploitable vulnerabilities are countered. Known operational vulnerabilities can be separated into two categories, situations where the environmental constraints in the security target are violated, and other ways in which the system may be configured or used insecurely. They are, however, relevant to the ease of use analysis, where it is necessary to show that the administrator is able to determine if the system or product is configured insecurely.

4.183 Having analysed each vulnerability, the (construction or operation vulnerability) analysis must consider the impact of each vulnerability in combination with all other vulnerabilities that may interact with it.

4.184 The vulnerability analysis must consider at a minimum all of the information contained in Figure 9 which is taken from ITSEC Figure 4. The verbs *state*, *describe* and *explain* are used to express the degree of rigour required for the correctness evaluation, as defined in Chapter 0. As the assurance level increases, the requirement for a semiformal and then a formal representation of the design, as defined in Chapter 0, is introduced.

	E1	E2	E3	E4	E5	E6
Security Target	S	S	D	D	E	E
Security Policy Model				D	E	E
Functions	S	S	D	DS	ES	EF
Architectural Design	S	S	D	DS	ES	EF
Detailed Design			D	DS	ES	ES
Hardware and Source Code				D	E	E
Object Code						E
User & Administrator Documents	S	S	D	D	E	E

Key:	S	-	State
	D	-	Describe
	DS	-	Describe Semiformal
	E	-	Explain
	ES	-	Explain Semiformal
	EF	-	Explain Formally

Figure 9 Requirements for information at various assurance levels

Considerations

4.185 On occasions, it may be difficult to distinguish between operational and construction vulnerabilities. A flaw in the design may mean that the system can be operated insecurely; as a short term expedient this may be redressed by operational procedures whilst the technical vulnerability is addressed. What is more important is that the vulnerability is actually identified in one or other of the two vulnerability analyses. These operational procedures must be included as assertions or guidance in the documentation provided by the sponsor (the security target or operational documentation).

Ease of Use Analysis

Description

4.186 This aspect of effectiveness investigates whether the TOE can be configured or used in a manner which is insecure but which an administrator or end-user of the TOE would reasonably believe to be secure. The ease of use analysis must identify all possible modes of operation of the TOE, including operation following installation, failure, and operational error and show that any human or other error that deactivates or disables security enforcing functions is easily detectable. If the TOE can be configured in an insecure manner, then this fact should be obvious to an administrator or end-user when in use.

4.187 The evaluator will validate the statements made in the analysis and will also perform additional testing to confirm or disprove the analysis. The installation procedure contained in the administration documentation will be followed to assess whether the TOE can be unwittingly configured insecurely.

Approach

4.188 The ease of use analysis could be divided into three sections. The first section could describe the approach taken, namely that the ease of use analysis will address whether the administrator would reasonably believe the TOE to be secure when it is in fact in an insecure state as identified in the operational vulnerability analysis.

4.189 The second section could identify all the possible modes of operation which have any effect upon the security of the TOE. For example, this could discuss, for a UNIX system, single user mode and multi user mode. It may cover maintenance modes, and normal operating mode. It should cover any special modes that may arise, for example, as a result of filling the audit trail.

4.190 The third section could analyse the security implications of the identified modes of operation, showing that in each mode either the TOE is secure (the security target is satisfied) or the insecurity is easily detectable.

Considerations

4.191 The ease of use analysis is closely related to the vulnerability analysis and the exact interpretation of an item as a construction vulnerability, an operational vulnerability, or an ease of use issue is often difficult to determine. It should be remembered that the effectiveness analyses required are intended to provide, between them, a complete vulnerability analysis for the TOE. It is not the intention of the ITSEC unduly to restrict the form of presentation, but rather to provide a set of tools to assist in the analysis.

UK Scheme Recommendations

4.192 The following sections propose the UK Scheme recommended approach to the ITSEC ease of use concept.

The Context of Ease of Use

4.193 ITSEC 6.31 defines ease of use as *an aspect of the assessment of the effectiveness of a TOE, namely that it cannot be configured or used in a manner which is insecure but which an administrator or end-user would reasonably believe to be secure.*

4.194 Some vulnerabilities cannot take effect unless a user makes a mistake. Most TOE users are assumed to be legitimate and will try to use a TOE securely. If a TOE is difficult or confusing to use, however, the users are more likely to make mistakes. A judgement has to be made, based on the motivations and skill levels of the various types of TOE user, and the usability of a TOE, concerning the likelihood of a user making a mistake which creates a security weakness. This analysis is called ease of use analysis.

4.195 The aim of the ease of use analysis is thus to demonstrate that the TOE cannot operate in an insecure state without the user being aware of this; otherwise an operational vulnerability will exist.

4.196 It is assumed, for the purpose of ease of use analysis, that the TOE's security enforcing functions, including their operational interfaces, are implemented and documented correctly.

4.197 An approach for the production of an ease of use analysis, which has been used successfully in practice, is given in the following sections. If an alternative approach is followed then it must involve a similar degree of rigour.

Primary Bases for Ease of Use

4.198 Ease of use analysis is concerned with security of operation. At all assurance levels, the key deliverables which should thus be used as the basis for an ease of use analysis are the security target and the correctness deliverables for operation.

4.199 All operational aspects for correctness (i.e.. user and administrator interfaces; delivery procedures; configuration, startup and operation facilities) are relevant to the identification of operational states.

4.200 Both *operational interfaces* and *operational procedures* are relevant to the identification of operational states.

4.201 *Operational interfaces* are supported by operational documentation, which gives guidance on when to invoke a particular interface and on the use of the interfaces. Also the interfaces themselves may offer a degree of self-documentation, e.g. in the form of prompts or accompanying help text. Examples of operational interfaces include:

a) user interfaces: these enable the end-user to interact with the TOE,

b) administrator interfaces: these allow administrators to perform privileged operations,

c) external application interfaces (e.g. APIs): these allow external applications to interact with a TOE,

d) communications interfaces: these allow messages and commands to be received from or sent to another computer system,

e) generation interfaces: these allow configuration options to be specified,

f) diagnostic interfaces: these report the results of diagnostic tests.

4.202 *Operational procedures* exist in the form of documentation alone. A delivery procedure is an example of an operational procedure.

4.203 Within the context of the TOE the required security is as specified by the security objectives, threats and security enforcing functions in the security target. The assessment of whether a particular operational state is insecure should thus be made with reference to the security target.

4.204 Where it is clearly evident that an operational interface or operational procedure is not related to the security of the TOE then it need not be explicitly mentioned within the ease of use analysis. However, all documentation applying to such operational interfaces and procedures must be referenced, in indicate, indirectly, which are asserted to be security irrelevant.

4.205 For all other operational interfaces and procedures relevant operational documentation must be considered and referenced in determining and analysing potentially insecure operational states.

Modes of Operation

4.206 In identifying operational interfaces and procedures the analysis must identify all possible modes of operation of the TOE. Examples of modes to be considered include:

a) Start Up,

b) Close Down,

c) Standalone / Networked,

d) Single-User / Multi-User,

e) Administrator,

f) Maintenance,

g) Failure.

4.207 For each operational interface and procedure potential errors in operation should be recognised. For each possible error consideration should then be given, by reference to the security target, as to whether the error could lead to a potentially insecure state. Where this is so it is then necessary to consider:

a) whether the operational documentation gives sufficient clear warning to prevent the error being made accidentally,

b) whether the TOE gives sufficient clear warning to prevent the error being made accidentally,

c) whether the TOE implements checks to prevent entry to the potentially insecure state,

d) whether the TOE, on entry to the insecure state, issues an alert, to the user or administrator, to clearly indicate that this has occurred,

e) whether the user or administrator is subsequently, before the insecurity could be inadvertently or maliciously exploited, required to invoke a facility which will detect and clearly report the existence of the insecurity.

4.208 Where one of the above factors applies, then the potentially insecure state will be of no concern (either because the TOE prevents entry to the potentially insecure state so that no insecure state will result, or because, whilst an insecure state might result, the user will be aware of this). Should none of the above factors apply however, then an operational vulnerability will exist.

4.209 The ease of use analysis must identify all potentially insecure states which could result from operational error. For each potentially insecure state it should then justify why the insecure state is of no concern; otherwise an operational vulnerability will exist.

4.210 It is acceptable for the ease of use analysis to focus on potentially insecure states. Thus, where it is clearly evident which potentially insecure states may result, this may be recorded without the need for the causal errors to be listed. However errors must be detailed where such detail is necessary to aid understanding of how a potentially insecure state results. Where it is consistently evident why a potentially insecure state is of no concern, this may be justified without reference to individual causal errors. However where such justification varies between different causal errors, then the different justifications must be made with references to associated errors or types of errors.

4.211 In certain cases a single error could lead to a number of potentially insecure states. However it might also be that none of the potentially insecure states is of concern because of a single factor. In such cases it is sufficient to specify the error and give a generic justification as to why it will not introduce an operational vulnerability; the potentially insecure states need not be listed.

4.212 Any arguments used in the ease of use analysis should reference relevant correctness deliverables, so that the analysis can be verified. In areas where the TOE is self documenting it may be appropriate to reference the TOE itself; however, in such cases, the developer should assure himself that the evaluators will be able to verify the ease of use analysis by operation of the TOE during testing.

Deactivation and Disabling Considerations

4.213 The deactivation or disabling of security enforcing functions and mechanisms represents one particular type of insecurity which might result from operational errors, and must thus be considered within the scope of the ITSEC Figure 4 deliverables. For the purposes of ease of use, security enforcing functions and mechanisms are essentially equivalent, mechanisms providing the abstract implementation of security enforcing functions. The emphasis should thus be on the loss of security enforcing functionality resulting from deactivation or disabling, with mention of mechanisms usually representing a simple exercise of correlation with security enforcing functions. This is particularly straightforward where there is a one to one mapping between a security enforcing function and a mechanism. However mechanisms must be considered further in the following cases:

a) where a one to one mapping does not exist between security enforcing functions and mechanisms,

   (The precise effect will be dependent on the nature of the mapping, but, for example, deactivation of a single mechanism might result in say the deactivation of more than one security enforcing function or the degradation of a single security enforcing function.)

b) where there are security relevant mechanisms which enforce binding (see paragraph 0).

   (Note that these are often low level, and as such may not feature explicitly in the security target.)

Configuration Considerations

4.214 The configuration of a TOE is the selection of one of the sets of possible features of the TOE. These features may be termed *configurable options*, and are typically exemplified by the privileges and protections for users, devices and files on the TOE.

4.215 The developer should be aware of the danger of delaying consideration of configuration until a TOE becomes operational. Experience suggests that it should be addressed at all stages of development. There is a danger that a well developed TOE may be rendered insecure by poor configuration.

Failure Considerations

4.216 For completeness, the consequences of hardware failures, e.g. power supply interruption or failure of a hardware component, must also be considered.

4.217 In certain cases it might be possible to easily identify the insecurities resulting from a particular hardware failure, e.g. if audit data is lost when the disk to which audit records are written fails.

4.218 In other cases however the consequences of failure may be more complex and a more general approach will be the most sensible. Typically this will involve confirming that users are made aware of the failure and that an appropriate procedure will then be followed to detect insecurities, e.g. involving examination of security parameters or restoring a trusted backup.

Assurance Level Considerations

4.219 Graduation, in accordance with the target assurance level, of the requirements for ease of use applies in a number of respects.

4.220 The increasing rigour of correctness specification, as required by the verbs *state*, *describe* and *explain*, forms the basis for the rigour of an ease of use analysis. Assessment of the completeness and clarity of operational specification must be made with reference to the requirements of the assurance level for rigour of the correctness deliverables.

4.221 The introduction of requirements for additional operational deliverables at higher assurance levels, e.g. in respect of recovery to a secure state after failure at E4 and above, increases the scope of operational correctness deliverables which must be considered by an ease of use analysis.

4.222 Deliverables additional to the security target and operational documentation must be considered in accordance with the requirements of ITSEC Figure 4. This applies to the disabling or deactivation of mechanisms. However the use of the additional deliverables may otherwise be limited to those areas where the security target and operational documentation provide insufficient detail to clearly demonstrate the consequences of a particular use of an operational facility, e.g. where it is unclear from operational documentation that a check exists within code to prevent entry to an insecure state.

Example Ease of Use Analysis

4.223 This example considers the UNIX *chmod* command. The *chmod* command allows users to change permissions on files. Users can grant read, write or execute access to themselves, to users in the group associated with the file, or to other users. For simplicity, only read, write and execute permissions are considered; use of the *chmod* command by an administrator is excluded from the example, as is consideration of start-up, close-down, single-user, maintenance and failure modes.

4.224 Consideration of the documentation for the *chmod* command reveals that three types of error are possible:

a) *Incorrect permissions* - the user specifies permissions that give users access which they should not have. This error leads to an insecure state.

b) *Incorrect file* - the user specifies an incorrect file so that permissions are not applied to the intended file and may instead be applied to another file. If the specified file exists the error leads to an insecure state.

c) *Syntactically incorrect input* - the user supplies arguments to *chmod* that are syntactically incorrect. UNIX consistency checking will detect the syntax errors and reject the command. Suitable error messages will alert the user to what has happened. An insecure state will not therefore result.

4.225 A single insecure state has been identified, which involves incorrect access permissions having been set for one or more files. For this example the insecure state is clearly evident and it will be sufficient for the ease of use analysis to record it without preliminary discussion of the causal errors. The error regarding syntactically incorrect input is a generic error which can be covered by a single generic paragraph in the ease of use analysis.

4.226 The insecure state, involving incorrect permissions on a file, can be detected in the following ways:

a) the file can be listed using the *ls -l* command and its permissions displayed (this facility can be used to check the permissions on a specified file)

b) a list of files that have been modified recently can be displayed via the audit facility (this facility can be used to detect files that have been modified unintentionally)

c) previous commands can be redisplayed using the *csh* history facility (this facility can be used to determine what file was specified or which permissions were requested in a recent invocation of *chmod*).

4.227 No operational vulnerability will exist provided that the operational documentation either:

a) gives clear warnings of the danger of applying incorrect permissions or applying them to an incorrect file, or

b) requires subsequent use of the facilities to detect an insecure state.

4.228 The ease of use analysis should give the appropriate justification as to why the insecure state is of no concern; otherwise an operational vulnerability will exist.

Chapter 5 Degrees of Rigour

Introduction

5.1 The ITSEC concept of degree of rigour provides a means for the level of detail in the correctness documentation to be altered according to the assurance level at which a TOE is to be evaluated.

5.2 Evaluation confidence is achieved by evaluators understanding the TOE through analysis of the information (evidence) provided by the sponsor and developer. The ITSEC requires greater rigour and depth in the evidence provided at higher assurance levels; this covers the description of the TOE given in the architectural and detailed design and in the implementation. The principle of increasing rigour is also valid for other forms of evidence and analysis.

5.3 There are three levels of detail at which the features of a TOE may be specified. In ITSEC terminology these levels are represented by a series of verbs which are referred to as the 'degrees of rigour'. The three levels of rigour are:

a) state (required at levels E1 and E2)

b) describe (required at levels E3 and E4)

c) explain (required at levels E5 and E6).

5.4 The degrees of rigour are defined in the Introduction to the ITSEC [Reference 0], paragraph 0.12 as follows:

a) state - means that relevant facts must be provided

b) describe - means that the facts must be provided and their relevant characteristics enumerated

c) explain - means that the facts must be provided, their relevant characteristics enumerated and justifications given.

5.5 When considering re-evaluation of a certified TOE to a higher assurance level, requiring an increase in rigour (e.g. from E2 to E3), a sponsor may need to revisit his whole approach to design documentation in order to provide the additional evidence required within each ITSEC deliverable. Thus, not only does the list of deliverables required increase with the assurance level, but the level of detail required within them also rises. The provision of information which is adequate for any future target evaluation level, rather than the initial one, should be considered during TOE development so as to avoid the need for costly rework.

Security Target

5.6 The security target for products requiring E1 or E2 evaluations must state all the relevant facts about the product, including what the security features and SEFs are. For an E3 or E4 evaluation it must also contain a description of how the features stated are provided. At levels E5 and E6 all the facts must be given including how and why the SEFs are provided.

Architectural Design

5.7 The architectural design is required at all assurance levels. As before, the documentation must maintain levels of rigour such that at levels E1 and E2 the relevant features are listed, at levels E3 and E4 their characteristics are also described (including how the features are implemented), and at levels E5 and E6 the reasons are also given.

5.8 As well as using greater formality, the content of the architectural documentation also increases with the assurance level. At all levels, the architectural design includes information on the general structure of the TOE, on its external interfaces, on any supporting hardware and firmware and on any security enforcing functions and protection mechanisms. There are also increasing requirements for modularity. The amount of detail regarding these aspects should therefore increase with assurance level, to meet the requirement for the appropriate level of rigour. In addition, and for the same reason, there should be an increase in the traceability information, showing how the SEFs are provided in the architectural design.

5.9 The low assurance levels are typically met by the provision of a functional block diagram, showing the major TOE components, with supporting statements, covering the aspects discussed above.

5.10 It is more likely that the high level output from a structured design methodology would be provided for the middle assurance levels; for example, a top-level SSADM diagram. However, supporting descriptions must also be provided, as the more formal specification alone is likely to meet the requirements for a statement rather than a description.

Detailed Design

5.11 The detailed design is required at assurance levels E2 upwards. As before, the documentation must maintain levels of rigour such that at level E2 the relevant features are listed, at levels E3 and E4 their characteristics are also described (including how the features are implemented), and at levels E5 and E6 the reasons are also given.

5.12 As with the architectural design, the requirements for formality also increase with assurance level, as do the requirements regarding the content of the detailed design documentation, in particular the requirements for specification of the interfaces between components and the requirements relating to modularity.

5.13 The remaining requirements relate to traceability of the security enforcing functions onto other levels of design, which also increase with the assurance level.

5.14 The level of rigour typically found in on-line technical manual pages generally forms a good basis for the requirement to state the relevant facts, but not to describe them. Hence, with minimal additional documentation, on-line manual pages are typically suitable for detailed design at assurance level E2, but not at higher assurance levels.

5.15 The various requirements are typically satisfied for the middle assurance levels by the provision of diagrams such as flow charts, entity life histories, etc with supporting descriptions.

Source Code, Hardware Drawings and Object Code

5.16 Source code and hardware drawings (as appropriate) are required at assurance levels E3 upwards; object code is required only at assurance level E6. The requirements for traceability from these items increase with assurance level.

5.17 It is possible that the requirements for traceability can be met by using information provided in the lowest level of the detailed design provided that there is a simple correspondence between the detailed design and the source code or hardware drawings. For example, the detailed design should identify the source code modules and functions involved with each SEF; the supporting information can be provided either in the low level detailed design, or in the source code comments.

5.18 The source code should contain sufficient comments for the evaluators to comprehend those aspects of the code that are not clear from the detailed design. This will contribute to meeting the requirements for rigour.

Operational Documentation

5.19 The requirements for user and administration documentation are unchanged through the assurance levels. However, the requirements for rigour do increase with assurance level.

5.20 In general, the operational documentation should provide guidance on the installation and configuration of the TOE, showing how to administer and use the TOE in a secure manner.

5.21 Figure 10 below provides a useful example of the effect of the degree of rigour on the user documentation and shows the transition from *state* to *describe*. The example is based on the UNIX chmod command.

Discretionary Access Control

This example provides details of the type of information that may be required in the User Guides, to describe a UNIX-style *chmod* command which allows a user to change the access permissions on a file. For simplicity, this discussion considers only read, write and execute permissions, and excludes from consideration the use of the command by an administrator.

Information provided at E1 and E2

At E1 and E2 a User Guide should identify the function of all commands available to the end user, stating that the *chmod* command is used to change the access permissions on a file. The User Guide should identify how each command should be used in a secure manner, covering the command syntax, and possibly giving some examples of its use.

Information provided at E3 and E4

At E3 and E4 a User Guide should also describe the process and effect of changing file access permissions. For example:

a) To change file access permissions, the command *chmod* should be used. The syntax of the command is:

   chmod (*filename, permission-list*) [/*echo*]

   where:

   filename is the name of the file for which the access permissions are to be changed;
   permission-list is the string describing the new access permissions in the following format (...);
   echo is an optional parameter allowing the user to choose whether to view the result of the *chmod* command. The default setting is /+echo (echo on), which causes the file name and access permissions to be listed on the screen. Setting /-echo (echo off) causes the display of the result of the command to be suppressed and replaced by a simple 'OK' response.

b) The possible responses to the *chmod* command are as follows:

   i) a file name and a list of permissions in the following format: (...)
   This indicates that the *chmod* command succeeded and that echoing was turned on.

   ii) the response 'OK'. This indicates that the *chmod* command succeeded and that echoing was turned off.

   iii) an error message. The text of the error message may indicate a syntax error, or that the user does not have sufficient privilege to change the access permissions for the file. (... or any other error ...)

Figure 10 'chmod' Command - Increasing Detail in Degree of Rigour (1 of 2)

Information provided at E5 and E6

At E5 and E6 the User Guide should explain the reasons for the security enforcing functionality. In addition to the text as described above, the User Guide might explain that:

a) The user may need to change the access permissions for a file which he owns, because:

   i) there may be changes in personnel or in the requirements for other users to see the file

   ii) the requirements for access permissions on a newly-created file may differ from those allocated by default.

b) The manner of specification of the filename parameter is consistent with the manner of specification of filenames in other commands.

c) The manner of specification of the access permissions is consistent with the listing provided by the *ls* command.

d) The results of the *chmod* command can optionally be echoed to the screen in the same format as that provided by the *ls* command. The option of a screen echo is provided in order to draw the user's attention to the effects of using the *chmod* command, so that the user can detect any unintended or insecure consequences of using the *chmod* command. For this reason, the echo option is selected by default, unless explicitly switched off. The *ls* command may also be used to check the access permissions for a file.

e) Use of the *chmod* command is an auditable event, in order to provide individual accountability for security-relevant actions (such as changing file access permissions).

f) It is recommended that the echo option is not switched off, in order to ensure that unexpected results of the *chmod* command are immediately detectable by the user.

Figure 10 'chmod' Command - Increasing Detail in Degree of Rigour (2 of 2)

5.22 Note that, dependent on the target audience for the user documentation, more detail than required by the ITSEC may need to be provided; for example, at E1 the developer may wish to explain various user aspects for inexperienced users as a condition of a development contract.

Effectiveness Documentation

5.23 The effectiveness criteria do not change by assurance level. However, some of the correctness documentation is used as input for the preparation of the effectiveness documentation and, as the assurance level affects the degree of rigour of the correctness documentation, hence the effectiveness documentation is influenced by the assurance level.

Chapter 6 Formality of Expression

ITSEC Requirements

6.1 Just as the requirements for rigour of expression increase with assurance levels, so do the requirements for formality of expression.

6.2 For assurance levels E1 to E3, it is sufficient to provide all documentation in informal style, i.e. in natural language. For evaluations under the UK Scheme, documentation should be provided in the English language. Foreign language documentation must be suitably translated. Although written in natural language, the documentation should not contain too many ambiguities and inconsistencies, as these may prevent the TOE from being implemented and used as intended.

6.3 For assurance levels E4 and E5, a formal security policy model is required. The architectural and detailed design documentation must be semiformal, as must the specification of security enforcing functions in the security target. A semiformal specification requires the use of a notation which is explicitly defined. It may be based on a restricted subset of the natural language. Alternatively, it may be based on accepted methodologies or diagrams, e.g. data flow diagrams, state transition diagrams, flow charts. Pseudocode may also be considered to be semiformal (provided that the rules and conventions are defined).

6.4 The ITSEC requirement for a semiformal style of expression can be met through a combination of styles and techniques; however, the documentation must be consistent, and the notations used must be defined. In addition, it should be noted that the evaluators will check, as required by the ITSEM, that the notation and styles used are capable of adequately representing the security features of the TOE.

6.5 For assurance level E6, the architectural design must be formally specified, as must the security enforcing functions in the security target. Formal specification is based on mathematical principles, and covers the syntax and semantics, assumptions and boundaries, as well as the proof itself. SEFs may be specified using set theory, or as part of the formal security policy model. There are a number of accepted formal notations in which the formal security policy model may be expressed, e.g. VDM, Z, GYPSY. Predicate logic and set theory may also be used, provided that the rules and conventions are defined.

Formal Model of Security Policy

6.6 In keeping with the increasing rigour and style of specification at higher assurance levels, a TOE must implement an underlying model of security policy at level E4 and above.

6.7 The model is an abstract statement of the important principles of security that the TOE will enforce. The model must be expressed in a formal style, i.e. written in a formal notation based upon well-established mathematical concepts.

6.8 While the formal model need not cover all the TOE's security enforcing functions (SEFs), the essential security characteristics should be modelled, i.e. the underlying security policy. For example, if the main purpose of a TOE is to provide access control, then all the access control functionality should be modelled, but other areas, such as audit, need not be modelled. For a TOE which claims F-B1, it is expected that the I&A, and access control parts of the TOE are included in the model, as a minimum.

6.9 However, an informal interpretation of the model in terms of the security target must be provided. This interpretation must correlate the model to the SEFs and show that:

a) the model does not contain any aspects of policy which are not completely reflected by one or more SEFs

b) no SEFs within the security target conflict with the policy within the model.

6.10 Note that the SEFs are specified both informally and semiformally (E4 and E5) or formally (E6), but that the interpretation is only directed to the informal SEF specifications.

6.11 It is unlikely that multiple models relating to one TOE would be practical, unless:

a) they are applied to totally unrelated areas of policy and functionality

b) additional modelling needs to be added to part of a published model (in which case all of the modified model should be included in the security target and it should use the same notation).

Non Published Models

6.12 When a model is used which has not been published, the formal consistency proofs must be correct and complete, and must be provided together with the model.

6.13 Producers of models should note that development of underlying formally specified models of security is impractical unless they have received substantial training or lengthy practical experience in the use of the chosen formal notation, and have adequate tool support.

6.14 The Certification Body may be contacted for advice on what models may realistically be provided.

Published Models

6.15 If a published model is used, all or part of the model may be referenced from the security target. If only part of a model is relevant, care should be taken to ensure that the subset of the model provides a coherent security policy.

6.16 ITSEC [Reference 0] paragraph 2.83 lists some examples of published formal models of security policy which cover various access control, integrity and data exchange policies.

Chapter 7 Traceability

Purpose of Traceability

7.1 The main objective of traceability is to ensure that the security enforcing functions (SEFs) identified in the security target have been correctly implemented in the TOE. A significant part of the evaluation involves checking the security functionality as it is refined through the development process to the final implementation. The aim is to ensure that functionality is provided to meet all the security objectives, and that vulnerabilities are not introduced along the way via additional supporting functionality.

7.2 Traceability involves the complete mapping by the developer of each SEF between different levels of representation in the TOE's documentation. It also considers the evaluated TOE in relation to the identified threats, security objectives and intended method of use. Traceability provides a significant part of the verification of the SEFs.

ITSEC Requirements

7.3 The following table summarises the ITSEC requirements for traceability:

Map From	Map To
Description of the architecture	Security enforcing functions defined in the security target
All levels of detailed design hierarchy	Security enforcing functions defined in the security target
Security mechanisms and functional units identified in the detailed design	Security enforcing functions defined in the security target
Tests	Security enforcing functions defined in the security target
Source code or hardware drawings	Basic components of the detailed design (Note, for E5-6, correspondence is to the functional units of the detailed design)
Security mechanisms represented in the source code or hardware drawings	Security enforcing functions defined in the security target
Tests	Security enforcing and security relevant functions defined in the detailed design
Tests	Security mechanisms represented in the source code or hardware drawings
User documentation	Security enforcing functions relevant to the end-user
Administration documentation	Security enforcing functions relevant to an administrator

7.4 Thus the ITSEC stipulates traceability from the majority of representations back to the security enforcing functions defined in the security target. In addition, the tests must be traceable to the detailed design and to the source code or hardware drawings. Also, the source code or hardware drawings must be traceable to the detailed design.

7.5 Each design level will contain more detailed descriptions of the functionality defined at the higher levels. The additional detail will allow the developer to further subdivide the functionality into the categories of security enforcing, security relevant and security irrelevant. Thus, as each level adds more detail, so the security enforcing aspect can be associated with a narrower portion of that detail. This is illustrated in Figure 0, shown earlier in this document.

Variation of Requirements with Assurance Level

7.6 Aside from the requirements for increasing rigour in the supporting text, the main difference at the higher levels is the need for increasingly more formal documentation. The ITSEC requirements for design documentation are summarised below:

Requirement	E1	E2	E3	E4	E5	E6
SEFs in security target	Informal	Informal	Informal	Informal & semiformal	Informal & semiformal	Informal & formal
Formal model	No	No	No	Yes	Yes	Yes
Correspondence from security target to model	No	No	No	Informal describe	Informal explain	Informal explain
Architecture description	Informal	Informal	Informal	Semiformal	Semiformal	Formal
How SEFs provided in architecture	State	State	Describe	Describe	Explain	Explain
Correspondence from architecture to formal model	No	No	No	No	No	Informal + formal explain
Detailed design description	No	Informal	Informal	Semiformal	Semiformal	Semiformal
How SEFs provided in detailed design	No	State	Describe	Describe	Explain	Explain
Source Code	No	No	Yes	Yes	Yes	Yes
Correspondence between source code and detailed design	No	No	Informal describe	Informal describe	Informal explain	Informal explain
Correspondence between source code	No	No	No	No	No	Yes

Requirement	E1	E2	E3	E4	E5	E6
and formal SEFs						
Test documentation	Optional	Informal	Informal	Informal	Informal	Informal
How SEFs tested	Optional	State	Describe	Describe	Explain	Explain
Correspondence between tests and formal SEFs	No	No	No	No	No	Explain
Correspondence between tests and detailed design	No	No	Describe	Describe	Explain	Explain
Correspondence between tests and source code	No	No	Describe	Describe	Explain	Explain

7.7 The requirements for formality are introduced at the higher levels, and relate to the design documentation.

Traceability for Deliverables not Developed for the ITSEC

7.8 It can be the case that TOEs are not developed with the ITSEC in mind, and it is often difficult to relate the TOE's design documentation to that described in the ITSEC. The structure of the ITSEC reflects the evaluation process, which by its very nature is a top down approach from requirements, through architecture and detailed design, to implementation. It is not assumed that a top down approach is used for development; any method can be used so long as the necessary documentation is provided. However, it may be easier to meet the requirements using a top down development style.

7.9 In practice, the developer may merge the requirements and architecture information into a functional specification. There may be multiple levels of detailed design, with some parts using informal text and some diagrammatic. There may be only a single level of detailed design including architectural information, details of interfaces and lower level design details. In some instances development passes straight to implementation from high level design with no intermediate low level design such as pseudo code.

7.10 TOEs produced to meet the TCSEC will usually have a Philosophy of Protection, which contains a similar level of detail to the security target but includes some effectiveness information, a high level architectural description, a high level design and interface details. At TCSEC B2 and above a Descriptive Top Level Specification (DTLS) must be provided, and many of the ITSEC architecture and detailed design requirements can be found within this Guide. The standardised, structured format of the DTLS is considered to meet many of the requirements for a semiformal interface description. Since traceability of security functionality is not addressed as explicitly for a TOE produced under the TCSEC scheme, this aspect usually requires additional work to meet the ITSEC requirements.

7.11 The need for additional traceability documentation will depend to a large extent on the original approach to design documentation. Where the design documentation contains sufficient detail to include the mapping information, for example, when a TOE is developed using SSADM, it will probably not require any separate mapping information, as the SSADM methodology ensures that each level of design relates to the level above and, where necessary, peer documents. Mapping can be included within other semiformal representations either by using diagram labelling techniques similar to SSADM, or by including references within the diagrams, such as function names.

7.12 Where the semiformal requirement leads to the production of an additional parallel design representation, this will not add to the assurance provided unless it is closely integrated with the existing informal documentation (for example by use of many detailed references), and can be viewed as adding clarity and structure to the existing documentation. Addition of a simple representation of major design aspects using diagrams will not meet the requirements.

Adequacy of Traceability Information

7.13 Checking of the developer's mapping should be a trivial exercise consisting of looking up a reference from one document to another; this should not involve tedious searching. If such searching is required, it is a sign that the traceability evidence is inadequate.

7.14 For example, if the mapping information comprises a function name, then looking up that function name in a lower design level should simply be a case of looking it up in the document's index, or finding it in the numerically or alphabetically ordered function descriptions; the function should not be buried in the midst of other information with no index reference. A good indication that the evidence is inadequate is when tracing of SEFs into the design is particularly time consuming, or the need arises for the evaluation team to consult the sponsor or developer for explanation.

Semiformal/Formal Requirements

7.15 The ITSEC explicitly identifies the following mappings as no more than informal in character:

a) correspondence between source code and detailed design

b) correspondence between source code and formal SEFs.

7.16 The ITSEC also states that at E6 a mapping from the architecture to the formal model must be made via a combination of formal and informal techniques.

7.17 There remains the following correspondence information:

a) from security target to formal model

b) how SEFs are provided in architecture

c) how SEFs are provided in detailed design

d) how SEFs are tested

e) correspondence between tests and formal SEFs

f) correspondence from tests to detailed design

g) correspondence from tests to source code.

7.18 Of these, a), b) and c) may require correspondence between semiformal or formal statements. Since these represent a description or explanation between semiformal or formal documents, it is likely that, by their very nature, they will be at least partially semiformal. If the mapping is not done with the same level of formality, then some assurance in the validity of the lower level representation will be lost.

7.19 Both a semiformal and a formal document can include additional explanatory informal text. However, production of additional semiformal or formal documentation after the TOE has been produced, simply to satisfy the evaluation criteria, and without it being incorporated into the development process, can give little or no assurance beyond that provided by the informal design. If mapping information refers between informal parts of the design, with no clear relevance to the more formal parts, there will be little evidence that the more formal parts have actually been used in the refinement of the security enforcing functions. Further, if formal notation is simply used to create formal interpretations of informal statements from the TOE's documentation, there is no guarantee that the resulting formal document is self-consistent, or that it is consistent with any other documentation produced in a similar manner.

7.20 Evaluation at E4 and above is likely to be successful only if the ITSEC requirements have been borne in mind throughout the development process and prior to embarking on formal evaluation.

Traceability Guidelines

7.21 Traceability needs to be considered at an early stage by the sponsor and developer to ensure that due account is taken of the requirements for consistency between the deliverables produced by the sponsor or developer.

7.22 The deliverables must show how the security enforcing functions defined in the security target are provided within each refinement of the design. Whether this is done as a direct one to one mapping between the specific level and the security target, or whether it is done via step-wise refinement, is immaterial, provided that the objectives are met and sufficient information is provided to form the basis of the evaluator's independent assessment.

7.23 Meeting the specific ITSEC correspondence requirements may be done indirectly. For example, the detailed design may be shown to correspond to the security enforcing functions defined in the security target using a cross-reference document which maps via the architectural design.

7.24 The mapping information must contain a similar level of detail to that required for the correctness and effectiveness documentation, i.e. at E1 and E2 a simple mapping statement is sufficient, at E3 and E4 there should be a description of the mapping, and at E5 and E6 additional explanation is required, with a full justification of the relationships claimed.

7.25 The mapping information should be similar in character to the representations being mapped. Some of the structured design approaches suggested in the ITSEC contain such information within each design level. No additional documentation should be necessary. Where a mapping between formal representations is required, a formal approach should be adopted to provide the required level of assurance.

7.26 As an example of what should be provided, consider the case of an E4 or E5 evaluation of a TOE for which TCSEC style design documentation has been provided. To meet the ITSEC requirements, the developer will also have been required to supply semiformal descriptions of the architecture and the detailed design. These may have been supplied in the form of high level architectural diagrams and low level pseudo code. In this case, there should be a clear relationship between the TCSEC informal architectural description and the semiformal high level diagrams; this relationship could either be obvious by the layout or numbering of the documentation, or could be explicitly stated by the developer in a mapping document. In addition, there should be a similar level of mapping information from the architectural documentation to the detailed design documentation.

7.27 The ITSEC mandates the information required; it does not specify how that information should be organised. It is the sponsor's and developer's responsibility to ensure that the information provided for the evaluation is complete, consistent and correct.

7.28 Suitable techniques include:

a) incorporating textual references in informal text

b) assembling tables

c) hierarchical structuring of documentation

d) automated tools (such as configuration control tools)

e) traceability provided by semiformal design methods.

7.29 For example, each security enforcing function may have been given a unique identifier, e.g. SEF 1, SEF 2 etc. In the detailed design a table could be included which identifies each security enforcing function in one column. In the next column, alongside each SEF, a reference to a section of the detailed design document could be included where there is a specification of a corresponding component or mechanism.

Chapter 8 Functionality Classes

Functionality vs. Assurance

8.1 The functionality of a secure system or product should not be confused with its assurance. The security functionality of a TOE is the logic which fulfils the purpose of the TOE's security enforcing functions (SEFs), whereas assurance is the degree of confidence that can be placed in the security of the TOE.

Functionality Classes

8.2 A functionality class is a predefined set of complementary SEFs capable of being implemented in a TOE. Functionality classes permit standardisation and provide sponsors with a shorthand way of expressing claims from the TOE.

8.3 Predefined functionality classes may be referenced for a variety of technical and commercial reasons. They may be claimed by sponsors or vendors when procurers are familiar with the Trusted Computer System Evaluation Criteria (TCSEC) [Reference 0], or to facilitate comparison between computer security standards, or they may be mandated by procurers who are required to meet particular computer security standards.

8.4 Example types of functionality class include high integrity requirements for data and programs which may be necessary in database TOEs; high requirements for the availability of a complete TOE or special functions of a TOE which may be necessary for TOEs that control manufacturing processes; and high requirements with regard to the safeguarding of data integrity during data exchange. Other functionality classes exist which are near equivalents to the US TCSEC criteria.

Use of Functionality Classes in Security Targets

8.5 If a sponsor wishes to claim a predefined functionality class, e.g. such as those in Annex A of the ITSEC, the claim should be stated under the specifications of the SEFs in the security target in the following format:

"The TOE shall implement all the security enforcing functions of functionality class F-nn as specified in Reference x."

where nn is the functionality class descriptor

and x is the reference source for the predefined functionality class.

8.6 A TOE may claim more than one predefined functionality class. A TOE may also have such additional SEFs as the sponsor wishes - subject to maintaining consistency with the claimed functionality class(es) and linking to one or more threats - these SEFs should be grouped under the usual SEF headings of I&A, Access Control, etc.

8.7 A functionality class must be referenced as a whole, and the TOE must be claimed to meet the class wholly by its own functionality, i.e. it is not acceptable for a database which relies on the underlying operating system for I&A functionality to claim conformance with F-C2, F-B1, etc.

8.8 It is advisable to reference functionality classes from the security target and not to include the verbatim particulars, unless the sponsor has a particular need to append information to elements of the class (e.g. for traceability purposes), in which case the reason for the inclusion should be stated in the security target.

8.9 A sponsor may wish to claim functionality additional to that in the functionality class. The additional functionality must be expressed separately from the functionality class claim under the normal SEF headings of Identification and Authentication, Access Control, etc. and meet the ITSEC security target requirements for the appropriate assurance level.

8.10 The functionality of ITSEC TOEs is not constrained by the assurance level. Sponsors may claim any combination of functionality class and assurance level.

Effect of Assurance Level

8.11 At E1 and E2, the SEFs must be stated. The informal style of the ITSEC example functionality classes is sufficient.

8.12 At E3 and E4, SEFs must be described. This requires the sponsor to provide more information than appears in the examples - in effect, the sponsor must describe how the SEFs are to be provided for the sponsor's particular TOE. This description must be traceable to the statements in the functionality class.

8.13 At E5 and E6, the SEFs must be explained. This explanation must be traceable to the statements in the functionality class.

8.14 At E4 and E5 a semiformal description of the SEFs is required in addition to the informal one; at E6, a formal description is required. These parallel descriptions must be traceable to the functionality class.

ITSEC Example Functionality Classes

8.15 Note that in the ITSEC [Reference 0]:

a) the predefined classes are examples and may lead to problems establishing traceability

b) classes F-C1, F-C2, F-B1, F-B2 and F-B3 do not guarantee that a TOE will comply with the TCSEC classes.

8.16 TOE-specific dependencies within the functionality classes must be identified, including generic terms such as *authorised user*, *subject* and *object*, which must be defined in the security target.

UK Scheme Example Functionality Classes

8.17 The UK ITSEC Scheme has expanded the definitions of the ITSEC F-C2 and F-B1 example functionality classes. This expanded documentation specifies:

a) the threats

b) the security objectives

c) the required security mechanisms

d) the security enforcing functions.

8.18 The UK F-C2 and F-B1 functionality classes also resolve some ambiguities and duplications present in the ITSEC example functionality classes.

8.19 For all evaluations performed under the UK Scheme, the UK F-C2 and F-B1 functionality classes should be used in preference to those defined in the ITSEC, [Reference 0].

8.20 The Certification Body should be contacted for the latest guidance on the example functionality classes and any TOE-specific dependencies.

Functionality Classes Appropriate to Compartmented Mode Workstations

8.21 When specifying security requirements for compartmented mode workstation (CMW) evaluations, it is recommended that use of the F-B1 predefined functionality class is considered, as this provides the core functionality which has been required in CMW evaluations to date.

8.22 While sponsors may wish to consider the content of the CMW requirements expressed in the paper *Security Requirements for System High and Compartmented Mode Workstations* (MITRE Corporation, November 1987), this paper must not be claimed in security targets. In particular, it should be appreciated that it contains assurance requirements (e.g. requirements relating to the architecture) in addition to functionality claims. It should be noted that the paper also contains both functionality claims in excess of those required by the functionality class F-B1 and environmental assumptions which may not be appropriate to all TOEs.

8.23 The security target for a CMW product must therefore claim one of the following:

a) a functionality class (F-B1 or higher)

b) a functionality class (F-B1 or higher) with additional security enforcing functions

c) security enforcing functions explicitly identified in the security target.

Additional Functionality Classes

8.24 Sponsors and developers are welcome to propose additional functionality classes for generic instances of TOEs. Further functionality classes may, in fact, be in preparation. The Certification Body should be contacted for further guidance on the suitability of the proposed classes and means of public dissemination.

Chapter 9 Penetration Testing

Purpose of Penetration Testing

9.1 The main objective of penetration testing is to ensure that the security enforcing functions (SEFs) identified in the security target actually work in the operational TOE. The aim is to ensure that functionality is provided to meet all the security objectives, that the security functionality works as given in the security target, and that there are no exploitable vulnerabilities in the operational system or product.

Location of Penetration Testing

9.2 Penetration testing is usually carried out at:

a) the development site, for a product

b) the operational site, for a system.

9.3 For a system evaluation, the evaluators will generally need access to the operational system, but where the system is in operational use, it may be possible to perform some of the tests on a development system.

9.4 It may be possible to arrange part or all of the penetration testing at other sites, if more convenient (particularly in the case of systems which contain sensitive 'live' operational data). Any such arrangement should be described in the Evaluation Work Programme. Possible 'other sites' where penetration testing can be carried out are:

a) the CLEF site

b) a third-party site, with the agreement of the third party.

Other Activities During Penetration Testing

9.5 The evaluators may use the penetration test visit to perform other activities, such as:

a) at any site:

- check the version number used for testing is consistent with that being evaluated
- check the correct configuration of the test setup
- repeat a sample of the developer's tests (ITSEC E2-6.13)
- perform additional tests to search for errors (ITSEC E1-6.13)
- check the use of the configuration control system (ITSEC E2-6.17)
- check the use of configuration management tools (ITSEC E4-6.17)
- check example audit trail output (ITSEC E5-6.17)
- check the operation of the startup and operation procedures (ITSEC E1-6.37)
- check diagnostic tests and results (ITSEC E2-6.37)
- repeat installation and configuration procedures (ITSEC 3.33)
- test to confirm or disprove the effectiveness analyses (ITSEC 3.24, 3.28, 3.33, 3.37)

b) at the development site only:

- check the physical and procedural security of the development environment (ITSEC E2-6.23)
- check the delivery and generation procedures (ITSEC E2-6.34)

c) at the operational site only:

- check the application of the delivery and configuration procedures (ITSEC E1-6.34).

ITSEC and ITSEM Requirements for Test Activities

9.6 This section covers only the requirements for test activities and does not include additional areas such as configuration control, installation, etc.

Coverage of Developers' Tests

9.7 The ITSEC requirements for developers' tests at the different assurance levels are detailed in Part II of this Guide. In particular it should be noted that the developer or sponsor may provide test documentation as an option at assurance level E1, but that test documentation is mandatory for all other assurance levels.

9.8 Each security enforcing function must be covered by at least one test in the supplied test documentation, and the test documentation must state the correspondence between the tests and the security enforcing functions defined in the security target.

9.9 At assurance levels E3 and above, the test coverage must be sufficient to allow the test documentation to describe or explain (as appropriate to the assurance level) the correspondence between the tests and:

a) the security enforcing functions defined in the security target

b) the security enforcing and security relevant functions defined in the detailed design

c) the security mechanisms as represented in the source code or hardware drawings.

9.10 At assurance level E2 and above, the ITSEM (paragraph 4.5.70) requires that every testable statement in the security target is used in a test.

9.11 At assurance level E3 and above, the ITSEM (paragraph 4.5.71) requires that every interface to every security enforcing or security relevant basic component is used in a test.

9.12 At assurance level E4 and above, the ITSEC (paragraph E4-6.11) requires that the test documentation also includes a justification why the test coverage is sufficient.

Regression Testing

9.13 The ITSEC (paragraph E3-6.12) states that at assurance levels E3 and above "evidence of retests after the discovery and correction of errors is obligatory to demonstrate that the errors have been eliminated and no new errors have been introduced".

9.14 The ITSEM (paragraphs 4.5.80 and 4.5.82) requires the evaluators to check all retesting following the correction of errors, stating that "the evaluators must review the developer's statements about regression testing in the test plan to ensure that adequate testing is performed. The evaluators must ensure that the regression testing strategy is followed". This set of requirements is not qualified in the ITSEM with a range of applicable assurance levels. However, the explanation at paragraph 4.5.81 of the ITSEM states the underlying principle that the corrections made should be consistent with the detailed design. This is not appropriate at assurance level E1, where there is no requirement either for detailed design documentation or for test evidence.

9.15 UK Scheme practice is that the evaluators will check regression testing where provided at assurance levels E1 and E2, and will require evidence of regression testing at levels E3 and above.

Check of Developer's Tests

9.16 The evaluators are required (by ITSEC paragraphs E2-6.13) to use the library of test programs to check by sampling the results of the developer's tests at assurance level E2 and above.

9.17 The evaluators will repeat some of the developer's tests, and will perform at least one additional test for each security enforcing function, where appropriate including both positive and negative tests. Paragraphs 4.5.83 and 4.5.84 of the ITSEM require that such additional tests must differ from the tests supplied by the sponsor.

9.18 At assurance levels E3 to E6, paragraph 4.5.84 of the ITSEM requires that the evaluators perform this additional testing for each security relevant function.

9.19 Paragraph 4.5.85 of the ITSEM allows the evaluators to specify these additional tests but to enlist the support of the sponsor in performing the tests. In such cases, the evaluators must witness the performance of the tests.

9.20 Paragraph 4.5.86 of the ITSEM requires the evaluators to verify that the actual test results conform to the expected test results.

Penetration Tests

9.21 The ITSEC requires (in paragraphs 3.24, 3.28, 3.33 and 3.37) the evaluators to carry out penetration testing to confirm or disprove the following effectiveness analyses:

a) the minimum strength of critical mechanisms

b) the exploitability of construction vulnerabilities

c) the ease of use analysis

d) the exploitability of operational vulnerabilities.

9.22 At assurance level E1, the evaluators are required (by ITSEC paragraph E1.13) to perform tests covering all security enforcing functions identified in the security target. However, the evaluators are not required to duplicate testing performed by or for the sponsor where adequate evidence of that testing is provided. The evaluators will check by sampling the results of any such tests.

9.23 The evaluators will also perform additional tests to search for errors, as required by ITSEC paragraphs E1-6.13.

9.24 The evaluators may also perform ad hoc tests in order to investigate unexpected results, as permitted by ITSEM paragraph 4.5.103.

Aims of Penetration Testing

9.25 The evaluators' aim, in performing penetration testing, is to demonstrate that the target of evaluation has no exploitable vulnerabilities (or, conversely, to demonstrate the exploitation of a vulnerability, where an exploitable vulnerability exists).

9.26 In order to satisfy the requirements of the ITSEM and of UKAS that the testing should be repeatable, reproducible and objective, the penetration tests and results will be recorded in detail by the evaluators. In addition, the configuration of the TOE (including version numbers) will be confirmed by the evaluators before and during testing.

Support and Facilities Required for Evaluators' Testing Activities

9.27 Before the evaluators visit a site, they will contact the sponsor and site management to make suitable arrangements for the visit, agreeing:

a) the date, time and venue for the visit

b) the accommodation and facilities to be used during the visit

c) the structure of the visit

d) the equipment required

e) any other activities to be carried out by the evaluators during the visit

f) the level of support required during the visit.

9.28 The following third parties may also attend site visits by the evaluators, in order to oversee or assess the evaluators' activities:

a) the Certifier

b) the UK Accreditation Service (UKAS), as part of the CLEF's accreditation requirements

c) the CLEF's Quality Assurance department.

9.29 The evaluators will require access to the TOE and test equipment where they are not overlooked and where they can work undisturbed. They may also require the use of standard office equipment (e.g. telephone, photocopier, secure storage).

9.30 The evaluators will require computer resources. The details of the resources required will depend on the nature of the TOE and on the assurance level. The resources required should be agreed prior to the visit. A typical set of required resources is as follows:

a) access to the TOE for the duration of the visit (note that for systems containing highly protectively marked data, the issue of the evaluators' access and clearances must be addressed before they are granted access)

b) dedicated access to the TOE for some or all of the visit

c) a set of ancillary equipment (e.g. magnetic media) for the TOE

d) facilities for installing and removing evaluation software and test results (if permitted by site regulations)

e) test data

f) developer's test tools, and any tools available from the sponsor or developer for evaluator-specified tests.

9.31 The testing may be structured in such a way that some of the equipment may need to be provided only for part of the visit. Any such arrangement should be agreed prior to the visit.

9.32 The evaluators will not be able to discuss their working methods or their results with the sponsor, developer or site management during the visit. However, any observations or problem reports raised during penetration testing by the evaluators may be communicated directly to the sponsor and/or developer, as well as to the Certifier, during the visit.

9.33 The facilities required for testing depend on the system or product under test. The developer's test programs, tools, procedures and results will be required (except in the case of an evaluation at level E1 where test evidence has not been provided by the developer). In addition, the evaluators may require other, more general, test equipment in order to carry out further testing, such as oscilloscopes, logic analysers or PCs. Any such requirements will be discussed between the evaluators and the sponsor prior to the visit.

9.34 If the sponsor or developer uses calibrated equipment for testing, or if the sponsor or developer supplies calibrated equipment for use by the evaluators during penetration testing, that equipment should be supported by current calibration certificates traceable back to a UKAS-accredited calibration laboratory. The evaluators may wish to take copies of the calibration certificates in order to satisfy UKAS requirements concerning the validity of test results.

9.35 Although the evaluators will require privacy for their work, they must be able to call on technical support from the sponsor or developer during testing.

9.36 Areas where the evaluators may require technical support include the following:

a) installation of the TOE

b) configuration of the TOE

c) familiarisation with the method of use of the system, product, test equipment or any combination of these

d) help with re-running some of the developer's tests

e) assistance with some of the more complex penetration tests, e.g. use of unfamiliar operating system facilities to load the system.

9.37 The evaluators may choose to ask the developer or sponsor to perform one or more tests, witnessed by the evaluators.

9.38 The amount of technical support required will depend on the nature and complexity of the TOE, but will typically amount to one person on standby throughout the visit with possibly one day's actual support effort.

Penetration Testing Disclaimer

9.39 The evaluators may require the developer or sponsor to sign, prior to starting penetration testing, a disclaimer to the effect that the evaluators will not be held liable for any damage which may occur during penetration testing. It will typically show the developer's or sponsor's confirmation that all information on the operational product or system has been backed up, and that the evaluators are free to change any information in any way.

Sequence of Events

9.40 A typical sequence of events for a penetration test visit is as follows:

a) introductory meeting

b) familiarisation with the TOE and test equipment

c) tour of the site (if it is a development site)

d) queries about the development environment (if the visit is to a development site)

e) check delivery and generation procedures

f) check configuration, calibration and versions of the TOE and of the test equipment

g) repeat a sample of the developer's tests

h) penetration tests, in the following order:

   - familiarisation
   - identification and authentication SEFs
   - access control SEFs
   - object reuse SEFs
   - accountability and audit SEFs
   - confirmation or disproof of effectiveness
   - 'crash tests'

i) additional tests to search for errors

j) ad hoc tests to explore unexpected results

k) 'wash-up' meeting.

9.41 A typical penetration test visit will start with a meeting to introduce the evaluation team to the TOE and to the facilities provided by the sponsor or developer. This meeting will also allow the evaluators to meet the team responsible for providing technical support during the visit.

9.42 The evaluation team will receive a briefing on the use of the TOE and of the test equipment provided. They will also perform any other site-based activities (such as checking delivery and generation procedures). These activities may involve interviewing staff and checking records held at the site. In the case of the activity "use the developer's tools to create selected parts of the TOE and compare with the submitted version of the TOE" (required at assurance level E4 and above) the activity will involve the evaluators rebuilding parts of the TOE using the developer's configuration management system.

9.43 The evaluation team will check that the TOE is the same version as is under evaluation, and that the test equipment is correctly calibrated.

9.44 The evaluators will check the configuration, calibration and versions of the equipment at the beginning of each session. If any new equipment is brought in for a session, then the developers may be asked to explain its use.

9.45 The evaluators will usually work in pairs to perform tests (including penetration tests), with one member performing the test while the other records the results. The evaluation team will work from test scripts prepared in advance, so that the tests are repeatable and reproducible.

9.46 The evaluators will usually repeat a sample of the developer's tests before performing penetration tests, in order to familiarise themselves further with the TOE.

9.47 The evaluators will work in private unless they specifically ask for assistance from the sponsor or developer. Such assistance may be in the setting-up of equipment, or in the performance of tasks requiring specific knowledge (e.g. manipulating an operating system) or high complexity (e.g. performing an action while another tester maintains a stream of traffic on a communications line).

9.48 The evaluators may hold a meeting with the sponsor or developer at the end of the visit, to discuss any matters arising from the tests. These matters may include:

a) problems

b) a summary of the test findings.

Problem Reports

9.49 If the evaluators find a problem during testing, they should issue a problem report (an Evaluation Observation Report or a Security Fault Notification, depending on how serious the problem is). Such a problem report may be issued either:

a) later from the CLEF

b) on the spot (which is more likely if the time to respond to the report is likely to be critical).

9.50 In the latter case (and particularly if a Security Fault Notification is to be issued) the evaluators will consult the Certifier before issuing the problem report.

9.51 On receiving a 'major impact' Evaluation Observation Report (EOR) or a Security Fault Notification (SFN), the sponsor or developer will be required to clear the EOR or SFN by performing the following steps:

a) fixing the problem, including any consequential changes to the documentation

b) repeating the relevant developer's tests

c) submitting the TOE and the affected documentation for further testing by the evaluators (in order to show that the problem has been corrected, and that no further problems have been introduced).

9.52 If the evaluators issue a 'minor impact' EOR, the sponsor or developer should take the steps listed above in order to clear the EOR. However, it may be possible to justify why a TOE may achieve certification even with a 'minor impact' EOR outstanding; the Certifier should be consulted for guidance.

9.53 The effective use of a suitable configuration management system may limit the corrective work required to clear a problem report.

Factors Affecting the Cost of Evaluators' Testing Activities

9.54 The evaluators' testing activities will depend on the complexity and on the assurance level of the TOE. Testing over a period of two weeks is typical.

9.55 At assurance level E1, the developer's test evidence is optional. Where the developer has not provided test evidence, full testing will be performed by the evaluators. Where the developer has provided partial test evidence, the evaluators will complete the testing. The extent and cost of the evaluators' testing activities at assurance level E1 therefore depend on the test evidence provided by the developer.

9.56 Where the arrangements for penetration testing involve significant travel or accommodation costs (e.g. if a testing site is in a remote location, or if it is abroad), appropriate budgets should be arranged.

9.57 The level of technical support required is described above. In addition, the sponsor or developer should allocate some staff time for meetings:

a) at the beginning of the visit for introductions to the TOE and other equipment supplied, and

b) at the end of the visit, if it is agreed to hold a 'wash-up' meeting to discuss problems and a summary of the test findings.

Chapter 10 Development Techniques & Tools

Introduction

10.1 This chapter provides guidance for the developer and sponsor on the following techniques and tools necessary for development of secure TOEs at applicable assurance levels:

a) quality issues

b) development tools

c) semiformal specification

d) formal specification.

Quality Issues

General

10.2 The ITSEC places no requirements on the sponsor or developer for accreditation to external quality standards; however, the ITSEC requirements for content, presentation and evidence for configuration control reiterate parts of standards such as BS EN ISO 9001.

10.3 An evaluation does not utilise the results of assessments to quality assurance standards, although many developers will already have a Quality Assurance Programme in place, perhaps even assessed to a standard such as BS EN ISO 9001. Other developers may be used to a less formal approach. Experience from previous developers suggests that the formality of the UK Scheme raises the quality of the TOE.

10.4 Many of the developer's activities for the ITSEC can be considered as fulfilling essential parts of a quality system. The visit to the development environment is used to determine whether or not the evaluation team can have confidence that the TOE and other deliverables are being produced by the developer to an appropriate standard of quality and consistency.

10.5 During the development environment visit, the evaluation team will make checks on the following quality documents, as relevant to the TOE's claimed assurance level:

a) Project management procedures

b) Quality assurance procedures

c) Technical assurance procedures

d) Design standards

e) Coding standards

f) Documentation standards.

Effect of Assurance level

10.6 The following lists illustrate many of the requirements which the developer must satisfy. Note that the requirements become more stringent at the higher assurance levels due to the degree of rigour and additional conditions applied.

10.7 At all assurance levels the TOE must be uniquely identified, and in particular the version used for penetration testing must be identified. If a TOE has undergone a series of 'patches' or 'releases', then the numbers of these patches or releases must be given in addition to the version number.

10.8 At level E1, a simple configuration list is required. This configuration list must identify the TOE by its version number.

10.9 At E2 and above:

a) a configuration control system must be used

b) unique identifiers must be applied to the TOE, its basic components (e.g. modules which can be compiled) and all documents including the manuals

c) the information on the configuration control system shall state/describe/explain (as appropriate) how it is used in practice and applied in the manufacturing process in accordance with the developer's quality management procedures.

10.10 At E3 and above:

a) evidence of retests after the discovery and correction of errors relevant to security is obligatory to demonstrate that the errors have been eliminated and no new errors have been introduced (when an error is revealed by a test, the process followed to eliminate the error and perform retesting must adhere to the developer's quality procedures including change and configuration control, and satisfactory evidence must be produced to the evaluation team)

b) an acceptance procedure must be used to ensure that the TOE's constituent parts are of adequate quality prior to their incorporation within the TOE (the procedure should cover each stage of building the TOE, software, firmware and hardware, and the acceptance of previously evaluated or COTS components)

c) a standard for any programming language used defining unambiguously the meaning of all statements used in the source code.

10.11 At E5 and above:

a) the information on the configuration control system and the integration procedure shall explain how they are used in practice and applied in the manufacturing process in accordance with the developer's quality management procedures.

Development Tools

General

10.12 Current IT development can involve a wide range of development tools. The ITSEC places few restrictions on the choice of tools used for the development of TOEs, but does mandate an increasing degree of automation and control over the development process as the assurance level rises.

10.13 At higher assurance levels, the developer is required to use a tool-based configuration control system. Figure 11 gives an indication of the types of tools required by each assurance level.

Assurance Level	Tools Required
E1	none
E2 and above	test tools
E3 and above	well-defined programming languages
E4 and above	developer's tools, tool based configuration control system
E6	object code analysis tools

Figure 11 Levels and Tools

10.14 The configuration management system should ensure that there is a clear, complete and accurate representation of the TOE at all stages in its life cycle. This representation should reflect all changes that have been made to the configuration.

10.15 A tool-based configuration control system should enforce a clearly described configuration management policy and should encompass:

a) the traceability of every modification of the TOE to an approved change request

b) the traceability from effect to cause of any malfunction in the TOE due to a change

c) analysis of the effect of changes on unchanged components

d) the definition of responsibilities for change control

e) the control of access to TOE software modules during their development

f) the synchronisation of implementing TOE changes and updating TOE documentation

g) the generation of any previous version of the TOE

h) the auditing of implemented control procedures

i) the auditing of the TOE status accounting procedures.

10.16 It is necessary to provide confidence that the TOE has been implemented in a controlled manner. All alterations to the TOE should be authorised and controlled to ensure that they do not adversely affect the ability of the security enforcing and security relevant components to enforce the system security policy or product rationale. The use of digital signatures may be helpful here.

10.17 If tools are used to generate information for inclusion within deliverables, e.g. proof checking or source code statement coverage, then the evaluation team should be given access to the tools and related databases, if required by the assurance level criteria.

CASE Tools

10.18 With the use of semiformal notation comes the advantage of being able to automate aspects of the design process using Computer Aided Software Engineering (CASE) tools. The use of CASE tools enforces the use of a well-defined notation. It is not uncommon for these tools to be used for the development of TOEs for which the use of a semiformal notation is not a requirement of the detailed design, e.g. levels E2 and E3.

10.19 When considering whether to use a CASE tool, the developer is recommended to consider if it provides the following attributes which may be required depending on the nature of the TOE or assurance level:

a) suitability for meeting the semiformal requirements for architectural (E4 to E5) and/or detailed design (E4 to E6)

b) support for traceability of the security enforcing components, etc. through the design

c) control of design complexity (for example by supporting hierarchical decomposition)

d) a degree of automated design validation

e) provision of reporting facilities enabling errors and omissions to be identified

f) access control and audit measures

g) easy integration with tool or paper-based configuration control systems.

10.20 Where a CASE tool is used, consideration should be given to either providing the evaluation team with a copy of the tool's database or, where the team do not have immediate access to the appropriate tool, access to the developer's tool. This could reduce the amount of evaluation work required.

Semiformal Specification

10.21 A semiformal style of specification requires the use of some restricted notation (or notations), in accordance with a set of conventions which are included in or referenced by the specification. The conventions are specified informally.

10.22 A semiformal style may be graphical in presentation, or based on restricted use of natural language. Examples of semiformal styles include data flow diagrams, state transition diagrams, entity-relationship diagrams, data structure diagrams, and process or program structure diagrams.

10.23 Program Description Language (PDL) or pseudocode is an acceptable semiformal notation for the purposes of low level detailed design specification provided that PDL standards exist which clearly define the syntax and semantics of the PDL constructs.

10.24 Structured design and development methods normally incorporate at least one such semiformal notation for requirements capture, together with prescriptive guidance on how to use the notation. Examples of structured design methods including such notations are as follows:

a) Yourdon Structured Method [Reference 0]

b) Structured Analysis and Design Technique [Reference 0]

c) Structured Systems Analysis and Design Method [Reference 0]

d) Jackson Structured Design [Reference 0]

e) Jackson Structured Programming [Reference 0].

10.25 When choosing a semiformal specification, the developer and sponsor should consider:

a) what features of the TOE need to be expressed in the semiformal notation (e.g. modularity of the detailed design)

b) whether the notation can adequately express these features.

Formal Specification

10.26 A formal style of specification is written in a formal notation based upon well-established mathematical concepts. The concepts are used to define the syntax and semantics of the notation, and the proof rules support logical reasoning. Formal specifications must be capable of being shown to be derivable from a set of stated axioms, and must be capable of showing the validity of key properties such as the delivery of a valid output for all possible inputs. Where hierarchical levels of specification exist, it must be possible to demonstrate that each level maintains the properties established at the previous level.

10.27 The syntactic and semantic rules supporting a formal notation used in a security target must define how to recognise constructs unambiguously and determine their meaning. Where proof rules are used to support logical reasoning, there must be evidence that it is impossible to derive contradictions. All rules supporting the notation must be defined or referenced. All constructs used in a formal specification must be completely described by the supporting rules. The formal notation must allow the specification of both the effect of a function and all exception or error conditions associated with that function.

10.28 Examples of formal notations are VDM, Z, the RAISE Specification Language, Ina Jo, the GYPSY specification language and the ISO protocol specification language (LOTOS). The use of constructs from predicate (or other) logic and set theory as a formal notation is acceptable, provided that the conventions (supporting rules) are documented or referenced (as set out above).

10.29 The use of formal methods, mandated by the ITSEC at higher assurance levels, sometimes causes problems for the developer because of its novelty. The following paragraphs provide guidance from the ITSEM [Reference 0, paragraphs 6.3.34 to 6.3.41] on selection of formal techniques and tools.

10.30 *Underlying Formal Specification and Description Techniques:* at level E6, the ITSEC requires a formal description of the architecture of the TOE and a formal specification of the security enforcing functions.

10.31 Following ITSEC E6 requirements, formal comparison can be made between the formal description of the architecture and the underlying formally specified model of security. This comparison is not always easy: formal techniques are currently employed mainly for describing and proving static properties of TOEs. In this case, a combination of formal and informal techniques becomes necessary.

10.32 Another comparison can be conducted between the formal specification of the security enforcing functions and their realisation in the TOE. A precise formal specification of the functionality of the TOE involves the use of a mathematical notation and is therefore abstract. It is a functional or semantic definition of what a system does, without stating how this should be accomplished. Given a formal specification of the functionality of the TOE, properties of the TOE can be formally stated and proved. It is also a precise standard for the implementation.

10.33 A formal style of specification is written in a formal notation based upon well established mathematical concepts. Most formal notations use the constructs of mathematical logic (predicate calculus and, more recently, modal logic) and set theory.

10.34 There are three complementary techniques or methods for formal description. Operational definitions use an abstract interpreter to define the TOE. These are the least abstract, and resemble implementations. Denotational descriptions map the TOE directly to its meaning. Axiomatic or equational definitions describe properties of the TOE.

10.35 It is recommended that the developer selects formal specification and description techniques based on the following considerations:

a) Levels: to allow the system designers or user to look at the formal description in as much or as little detail as desired, the description should be split into levels ranging from the top level flow of control to the details of each operation.

b) Modular: all but the top level of the formal description should be modular. This will enable the design of each operation to be considered in isolation.

c) Concise: the notation should allow the necessary concepts to be expressed in a concise manner. A notation which is clumsy or verbose will unnecessarily lengthen the description.

d) Understandable: the formal specification notation should not be difficult to understand.

e) Abstract: the formal description should not dictate any issues which need not be resolved until the implementation stage. Although the top level flow of control is vital to the design of the system, it is often the case that, at lower levels, the ordering of certain events is immaterial.

f) Sound: to allow formal proofs of correctness to be carried out, the description technique should have a sound mathematical basis

10.36 *Formal specification tools:* briefly, these are defined as tools which implement - and techniques which use - mathematical logic. These techniques and tools aim to provide conclusive proof that the mathematical expression of a TOE strictly satisfies its mathematical specification. However, this does require correct conversion of the specification and proposed implementation into mathematical notation.

10.37 Formal methods supported by a tool shall, more precisely, be defined by means of:

a) formally specified syntax and semantics for notations used

b) algorithms to manipulate formulas of the languages

c) a set of proof rules by which correctness (completeness and lack of ambiguity) of the specification may be inferred

d) a framework for refining a specification into a concrete implementation.

10.38 *Expressiveness of Formal Specification Languages* employed by a tool shall be sufficient to formally describe the security policy and the components of an IT system enforcing that policy, e.g. in terms of invariant predicates. For the formal specification language there shall be concepts to structure the design specification in hierarchically ordered specification levels to refine a design specification from top-level TOE specification down to low-level program specifications.

Chapter 11 Integrating Certified Products

Introduction

11.1 As the range of certified products increases, the topic of integration is becoming more important to the development of secure TOEs.

11.2 While significant benefits can accrue from the use of Commercial Off-The-Shelf (COTS) products which have already been successfully evaluated and certified (e.g. lower risk, lower cost and timely availability), integrators should be aware of the security issues involved when designing TOEs from certified products.

11.3 The following sections provide general guidance on this topic. As the nature of integrated TOEs can be complex, integrators are advised to contact the Certification Body for further guidance.

Composite TOEs

11.4 As evaluated products become more widely available, it is likely that they will be used as building blocks for composite products or systems. For instance, a secure system might consist of a special purpose evaluated application using an evaluated database management system and running on an evaluated operating system.

11.5 The following terms are relevant to the evaluation of composite TOEs:

a) a *certified component* is a system or product which has previously been certified and is now being used as a component of a larger TOE

b) a *composite TOE* is a TOE which contains a number of components, at least one of which is a certified component

c) an *uncertified component* is a component of a composite TOE which is not a certified component.

11.6 Evaluation of composite TOEs can apply in a number of different cases:

a) installing a system consisting solely of a single certified product

b) composing a number of certified products into a composite TOE

c) linking a number of certified systems together

d) producing a TOE which consists of a mixture of certified and uncertified components.

11.7 Case a) above is an issue for the accreditor of the system, or the individual taking responsibility for accepting its use, and is outside the scope of this Guide. Should the accreditor require that the TOE must be evaluated then this would be a special occurrence of case d).

11.8 Although cases b) to d) are technically similar to the situation where a TOE design involves only uncertified components and all the correctness and effectiveness deliverables have to be provided, the concept of composite TOE evaluation enables previous evaluation results to be used and so limit the scope of the deliverables that need to be supplied.

11.9 The following deliverables are required for an evaluation of a composite TOE:

a) certificates/certification reports for the certified components

b) correctness deliverables for the uncertified components

c) some correctness deliverables for the composite TOE as a whole, to cover the integration of the various components (e.g. system testing)

d) effectiveness deliverables for the composite TOE as a whole

e) any correctness or effectiveness deliverables for the certified components which are required to support the argument given in the effectiveness deliverables for the composite TOE (in many cases, these will not be required).

Correctness

11.10 Some correctness deliverables for the composite TOE as a whole will be required, even if the TOE consists entirely of certified components. This is necessary to ensure that the integration of the various components correctly preserves the security objectives of the certified components. The following list gives a general idea of the type of information that will be required for each ITSEC requirement:

a) a security target for the TOE

b) architectural information for the TOE, showing its overall structure, how the components fit together, dependencies between components, separation, and how and where the TOE's SEFs are provided

c) architectural and detailed design information for any uncertified security enforcing or security relevant components

d) detailed design information for the TOE, covering details of the interfaces used between the components, any configuration options chosen for individual components, and how the SEFs and mechanisms are provided

e) source code or hardware drawings for any uncertified security enforcing components

f) tests of the TOE, covering all the SEFs and the integration of the various components

g) a configuration list for the TOE

h) details of the configuration control of the TOE, including control of the certified and uncertified components, and TOE documentation

i) details of the tools used, and languages and compilers used, and their options

j) security information, covering protection for the integrity of certified and uncertified components, and confidentiality of documentation

k) operational documentation for the TOE, including details of the documentation to be provided for the certified components, and guidance for secure startup and operation

l) generation and configuration procedures

m) delivery procedures.

Effectiveness

11.11 Effectiveness differs from correctness in that effectiveness applies to a TOE as a whole and cannot be partitioned in the same way as correctness. Therefore, the effectiveness deliverables for a composite TOE must be provided for the TOE as a whole, as would be the case for a TOE whose components were all uncertified.

11.12 In addition, where any of the analyses for the TOE rely on evidence provided in one of the component's analyses, then the documentation for that analysis must be provided as supporting evidence.

Chapter 12 Previously Evaluated & Non-Standard TOEs

Introduction

12.1 This chapter discusses the evaluation of previously evaluated and non standard TOEs as follows:

a) As certified TOEs evolve into new versions, due to modified security functionality, there may be a need to re-evaluate them to maintain their certification status.

b) Assurance profiles occur when a higher degree of confidence is required in some components than in others. This results in a TOE consisting of a number of components evaluated to different assurance levels.

c) Up-rating is a form of re-evaluation, in which the sponsor requires an increase in the target assurance level of a certified TOE, and no other change in the functionality or claims is made.

d) There may be circumstances when a sponsor wishes to limit the scope of an evaluation, (e.g. an ITSEC certificate is not required; the sponsor may wish to gauge more accurately the work involved for a large TOE by having it evaluated in a modular fashion)

e) The Certificate Maintenance Scheme is a way of maintaining a TOE's certification status as it is upgraded, without the immediate need for full re-evaluation, and hence reducing the cost of maintaining an ITSEC certificate.

Re-evaluation

Background

12.2 As existing secure products and systems evolve, it will become necessary to make changes to certified products and systems. For instance, a new log on mechanism may be added to a secure operating system, faults might be corrected, or a product might be made more efficient by recompiling its source code using a different compiler. Alternatively, the product or system could be left unchanged, but its threat, environment or intended method of use could be altered.

12.3 If such changes are made to a certified product or system, it is first necessary to determine whether the certificate can remain valid without extra evaluation work. This analysis is called *impact analysis*, and its production is the responsibility of the sponsor.

12.4 If the impact analysis determines that the certificate will or has become invalid after the change(s), an evaluation of the change(s) will be necessary if the certificate is to be renewed. This is termed *re-evaluation*.

12.5 In practice, impact analysis is often performed by a CLEF, usually under contract to the sponsor to provide consultancy on the requirements for re-evaluation and the extent of any actual re-evaluation work.

12.6 While a re-evaluation could be carried out by performing all the ITSEC correctness and effectiveness activities, efficient re-evaluation makes maximum use of the results of previous evaluation(s).

12.7 In particular, sponsors are encouraged to consider re-evaluation issues during the initial evaluation and so maximise their investment in the certification of the product or system. For example, sponsors should decide whether to pay for the necessary work to be undertaken to enable the categorisation of the TOE components to be documented in Chapter 7 of the Evaluation Technical Report (ETR). This chapter of the ETR would provide the basis for any future impact analysis of changes made to any construction phase or aspect, or operational aspect of the TOE. Reference would be made to the appropriate evaluation deliverables and would cover the following aspects:

a) for each of the construction phases examined, all parts of the TOE would be classified into one of security enforcing, security relevant or security irrelevant

b) the security relevant development tools used during the production of the TOE would be identified

c) any way in which the constraints or assumptions of the evaluation could impact re-evaluation or reuse of the TOE would be stated

d) any lessons regarding evaluation techniques or tools that would be useful for a re-evaluation would be stated

e) all archiving details necessary for the evaluation to be restarted would be stated

f) any specific recommended skills that subsequent evaluators should possess, before re-evaluation occurs, will be stated

g) the evaluators understanding of any way that the TOE could be configured such that it becomes insecure would be identified.

Deliverables

12.8 The deliverables needed for a re-evaluation are:

a) Deliverables from the previous evaluation (as required). These should be available if the sponsor is the same as for the previous evaluation, but in some cases, only a few deliverables from the previous evaluation will actually be required.

b) The certificate and certification report from the previous evaluation. The previous sponsor or the Certification Body should be able to supply these.

c) The ETR from the previous evaluation. Release of this document requires the agreement of those holding copyright and/or intellectual property rights, e.g. the CLEF, sponsor and Certification Body from the previous evaluation. Where the organisations involved in the re-evaluation are the same as those for the original evaluation there should be no problem obtaining the ETR; where any are different, it may not be possible to obtain the ETR. If the optional relevant information has not been recorded in the ETR's Chapter 7, its usefulness will be diminished.

d) Correctness deliverables for all changed components.

e) All changed correctness deliverables for the TOE as a whole (e.g. test documentation), as required.

f) All changed effectiveness deliverables, as required.

g) All other changed deliverables (e.g. developer's security procedures).

Correctness

12.9 All correctness deliverables affected by the change will need to be provided to the evaluation team.

Effectiveness

12.10 Whenever a change is made to the security features in the TOE's correctness deliverables, the effectiveness deliverables will need to be reviewed.

12.11 In the following list, the term *figure 4 deliverables* refers to the set of representations upon which, according to Figure 4 of the ITSEC [Reference 0], effectiveness analyses should be based. The following guidance is suggested for most cases; however, the Certification Body should be contacted for specific guidance:

a) If the security target is unchanged, the suitability analysis does not need to be repeated.

b) If the figure 4 deliverables are unchanged, the binding analysis does not need to be repeated.

c) If the figure 4 deliverables are unchanged, and no new insecure states are found, the ease of use analysis does not need to be repeated.

d) If the figure 4 deliverables for the critical mechanisms, and the environment in which they are used, are unchanged, and no new critical mechanisms have been introduced, the strength of mechanisms analysis does not need to be repeated.

e) If any of the correctness deliverables have changed, the construction vulnerability assessment will have to be repeated.

f) If the operational TOE or any of the correctness deliverables have changed, the operational vulnerability assessment will have to be repeated.

g) Penetration testing will always have to be repeated.

Assurance Profiles

Background

12.12 Assurance profiles, mentioned in paragraphs 1.21 to 1.22 of the ITSEC [Reference 0], occur when a higher degree of confidence is required in some components than in others. This results in a TOE consisting of a number of components evaluated to different assurance levels.

12.13 Selecting an assurance profile, like selecting an assurance level in a security target, is an issue for the sponsor and is outside the scope of this Guide.

12.14 An assurance profile evaluation should be regarded as a set of separate evaluations. These evaluations are referred to below as *sub-evaluations* even though each one is a full ITSEC evaluation and may have a separate security target. The set of security enforcing and security relevant components (or trusted computing base - TCB) for each sub-evaluation is those components which are targeted at the same level as the sub-evaluation, plus those components which are targeted at higher levels.

12.15 Each component of the TOE for a sub-evaluation should be treated as follows:

a) components targeted at a level below that for the sub-evaluation (if any) should be regarded as security irrelevant

b) components targeted at the same level as the sub-evaluation should be regarded as uncertified components

c) components targeted at a level above that for the sub-evaluation (if any) should be regarded as certified components - these components are referred to below as *higher-level components*.

Deliverables

12.16 The following deliverables are required for each sub-evaluation:

a) correctness deliverables for the uncertified components of the sub-evaluation

b) correctness deliverables (e.g. test documentation, security target, architecture) covering the sub-evaluation TOE as a whole

c) effectiveness deliverables covering the sub-evaluation TOE as a whole

d) correctness deliverables for the higher-level components of the sub-evaluation, where required to support the effectiveness deliverables for the sub-evaluation.

Example

12.17 This example illustrates the application of an assurance profile.

12.18 Consider a TOE with an E3/E4/E5 assurance profile. Three separate sub-evaluations are required, as follows:

a) an E5 sub-evaluation which would regard E3/E4 components as security irrelevant

b) an E4 sub-evaluation which would regard the E5 components as certified and the E3 components as security irrelevant

c) an E3 sub-evaluation which would regard the E4/E5 components as certified.

12.19 Figure 11 shows the overall architecture of the TOE schematically.

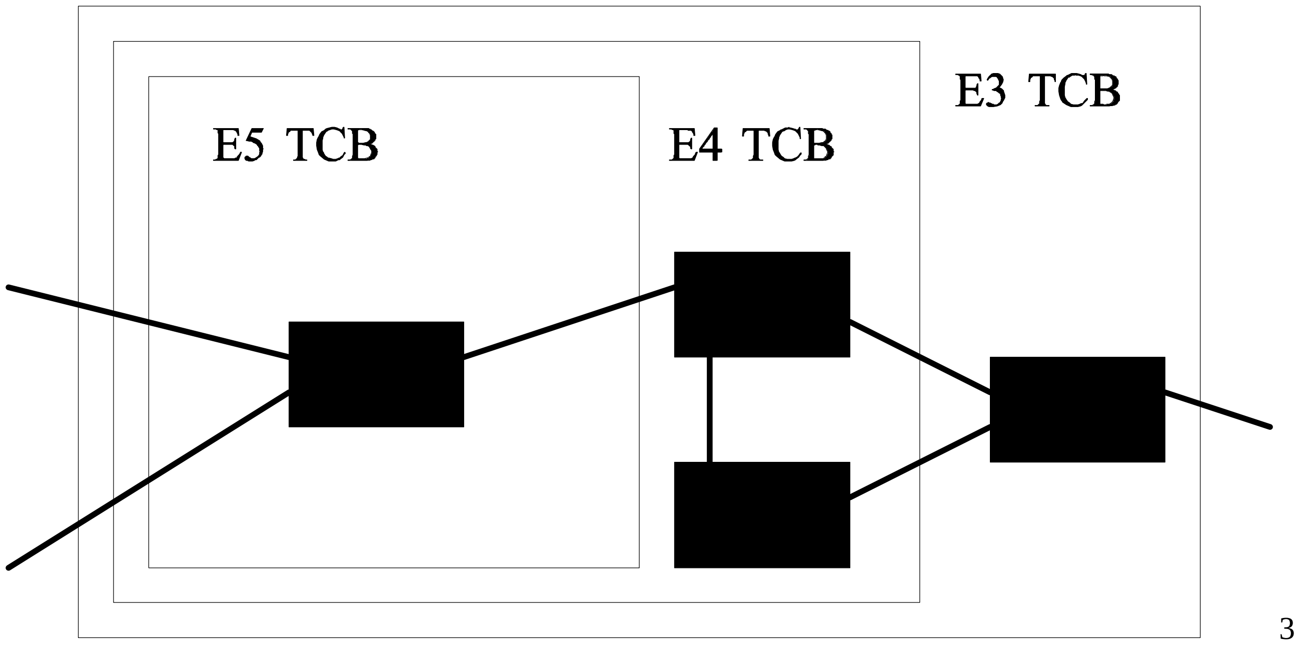


Figure 11 Overall TOE Architecture

12.20 The overall architecture of the TOE consists of a component, A, targeted at E5; components, B and C, targeted at E4; and a component, D, targeted at E3.

12.21 Consider the E4 sub-evaluation in detail. The E4 sub-evaluation TCB consists of components A, B and C. A is treated as certified. D is security irrelevant.

12.22 The required construction - development process deliverables for the E4 sub-evaluation would include:

a) A security target, describing the security objectives for which E4 assurance is required, and specifications of the SEFs which will meet them. Where an E4 objective is met by an E5 SEF, the E5 SEF could be documented by a reference to the E5 security target.

b) A formal model, and model interpretation, covering the E4 SEFs.

c) An architecture of the E4 TOE, covering components A, B and C in detail. The detailed architecture of component A could be covered by a reference to the E5 architecture deliverables. Separation between the E4 TCB and component D must be shown.

d) A detailed design for the E4 TOE which would cover components B and C. Component A need not be included because it is covered by the E5 detailed design.

e) Test documentation for the E4 TOE. This should contain:

- tests of the E4 TCB (components A, B and C) against the E4 security target
- tests of components B and C against the E4 detailed design
- tests of components B and C against the E4 source code
- the library of test programs and tools used in performing the above tests.

f) The source code for E4 components B and C, with a description of its correspondence with the detailed design.

12.23 It is likely that the development environments for the E3, E4 and E5 sub-evaluations will be identical. If this is the case, most of the *construction - development environment* deliverables produced for the E5 sub-evaluation can simply be referenced for the E4 sub-evaluation. The only exception is the *configuration list identifying the version of the TOE for evaluation*, which would cover components B and C, and could reference the E5 configuration list.

12.24 The *operation* documentation should cover the use (by users and administrators), delivery, configuration, startup and operation of components B and C. This could be done in two ways:

a) by providing separate documents for these components

b) by producing a single set of *operation* documents covering the whole TOE, and then identifying those parts of the documentation which are relevant to components B and C.

12.25 The following effectiveness deliverables would be required for the E4 sub-evaluation:

a) The suitability analysis should show that the combination of E4 and E5 SEFs is capable of maintaining the E4 security objectives.

b) The binding analysis examines the interactions between and within components A, B and C, and shows that the E4 TOE forms an integrated and effective whole.

c) The strength of mechanisms analysis shows that any critical mechanisms in components B and C meet the claimed rating of the minimum strength of mechanisms in the E4 security target. This assumes that the E5 security target claims at least as high a minimum strength of mechanisms as the E4 security target.

d) The construction and operational vulnerability analyses show that known vulnerabilities in components A, B and C do not violate the E4 security objectives. Note that the E5 evaluation showed that the known vulnerabilities in component A did not violate the E5 security objectives, but did not address the E4 security objectives.

e) The ease of use analysis will consider the modes of use of components A, B and C, and show that any insecure states (with respect to the E4 security objectives) are readily detectable.

Up-rating

Background

12.26 Up-rating is a form of re-evaluation, in which the sponsor requires an increase in the target assurance level of a certified TOE, and no other change in the functionality or claims is made.

Deliverables

12.27 The following deliverables are required for up-rating:

a) deliverables provided for the previous evaluation, updated to take account of:

- any additional requirements for content, presentation and evidence
- the transition from informal to semiformal to formal notations
- the transition for state to describe to explain
- Figure 4 in the ITSEC [Reference 0]

b) any additional deliverables required for the new assurance level

c) the certificate, certification report and ETR for the previous evaluation.

Examples

12.28 Some hypothetical composite TOE and re-evaluation situations are given below to show how up-rating can be applied.

An E3 System Evaluation

12.29 Consider a system which is required to be evaluated to assurance level E3. The system consists of an operating system previously evaluated to E3 and a special-purpose product previously evaluated to E2.

12.30 The evaluation could be performed as an up-rating to the E2 product to bring it to level E3, and then a composite TOE evaluation performed to show that the operating system and product together satisfy the security target for the system. Note that this approach would produce two certificates, one for the product and one for the system.

12.31 However, it may be more economical to perform only the E3 correctness actions for the product, and the effectiveness actions for the system as a whole. This would result in only one certificate, which would be for the system.

A Product Assurance Profile

12.32 Consider a product (say, an accounting package) which is required to be evaluated to E2. The product requires an operating system and communications package in order to run. The operating system and communications package have been evaluated previously to E1. It is possible to up-rate the operating system but not the communications package.

12.33 The sponsor could agree to adopt an assurance profile, with the accounting package and operating system being targeted at E2 but the communications package targeted at E1.

12.34 The evaluation would be performed as two sub-evaluations, targeted at E1 and E2.

12.35 The E2 sub-evaluation would involve the full correctness evaluation of the accounting package plus an up-rating of the operating system from E1 to E2. Then the effectiveness of the accounting package and operating system together would be evaluated.

12.36 During the E2 sub-evaluation, no assumptions could be made about the correctness or effectiveness of the communications package.

12.37 For the E1 sub-evaluation, the correctness of the communications package is presumed because it is a certified product, and the correctness of the accounting package and operating system is presumed because they are being evaluated to E2. The sub-evaluation would concentrate on the effectiveness of all three components.

Upgrading a System

12.38 Consider a system, previously evaluated to E4, which is to be upgraded by the addition of a communications link.

12.39 The following changes to the system need to be made to accomplish the upgrade:

a) an uncertified communications package is to be installed on the system

b) some modifications may need to be made to the log on component (these changes do not necessitate any change to the security target)

c) the security target should be changed to describe the security requirements for the communications package.

12.40 For the certificate to remain valid on the changed system, the developer needs to do the following:

a) produce correctness deliverables for the communications package

b) update the correctness deliverables for the log on component

c) perform testing for the system as a whole, concentrating on the communications and log on functionality

d) update the effectiveness deliverables for the system; most of the changes required will be related to communications.

Limited Evaluations

12.41 On rare occasions, a sponsor may not require a TOE to receive an ITSEC certificate, but may want only a limited set of evaluation activities to be performed. Before any agreements can be reached with a CLEF, or any work performed, the approval of the Certification Body must be gained for such a limited evaluation.

12.42 The reasons for performing a limited evaluation may be due to a lack of suitable deliverables or the fact that the TOE was never designed with the ITSEC in mind. The TOE may be an old system, sometimes referred to as a legacy system. The results of a limited evaluation can be reused as input to a full evaluation, when the limitation has been removed, to provide a full ITSEC certificate.

12.43 In this instance, the CLEF will prepare an Evaluation Work Programme detailing the reduced scope of the evaluation, which the certifier will assess it to ensure that the work proposed will form a practical and consistent set of evaluation activities.

12.44 The remainder of the evaluation will proceed as for a full evaluation within the agreed scope. On receipt of the Evaluation Technical Report, the certifier will issue a certification report detailing the results of the limited evaluation. However, no formal certificate can be issued for evaluations which perform a subset of the ITSEC requirements.

Certificate Maintenance Scheme

12.45 In recognition of the fast changing nature of systems and the frequency with which products are modified and upgraded, the Certification Body has introduced a Certificate Maintenance Scheme (CMS). The CMS provides sponsors with a practical means of maintaining the validity of a TOE's certificate and, therefore, their investment in the evaluation and certification.

12.46 Further details on the CMS are provided in UKSP 16, *UK Certificate Maintenance Scheme*, available from the Certification Body.

Annex A Example Security Target - LOCKPC

LOCKPC SECURITY TARGET

1 Introduction

This document contains a specification of the security enforcing functions of LOCKPC and a description of the environment in which LOCKPC will operate. The document forms the security target against which LOCKPC will be evaluated.

The version of LOCKPC to be evaluated is 1.0.

Note that unique numbers are assigned to the statements regarding method of user, environment, objectives, threats, security functions and mechanisms. This is for the purpose of explicit references in related evaluation documentation.

2 LOCKPC - Product Rationale

2.1 Introduction

The LOCKPC product is a combination of a hardware key, which prevents unauthorised access to data resident on a PC's hard disk and software which encrypts and/or decrypts information on that hard disk. LOCKPC can be configured to protect areas of the hard disk, allocating a logical disk drive to specified users identified by the physical key inserted in the PC's parallel printer port.

2.2 Intended Method of Use

U1 LOCKPC will be used to prevent unauthorised access to data on the host PC's hard disk by the use of a physical key in conjunction with software which carries out the encryption and decryption of information. The absence of either will prevent access to secure data.

U2 LOCKPC may be configured to protect separate parts of the hard disk, with practicality being the only real limit on how many different areas can be protected. Each of these secure areas will be accessed as though it were a different disk drive. The initial installation procedure will allow protection of up to four secure areas (as different logical disk drives), but an experienced user could easily increase this if they so wish. Despite this potency, it is expected that most users will only need to protect a few data areas.

U3 If more than one user is using a machine, then it is a simple task to set up several secure areas, with each user having a differently coded key. In this way each user may be using common software, but the data or personalised application cannot be accessed by other users.

U4 Each secure area is allocated a device name, this is synonymous with a volume label for a disk drive. The device name must be a valid DOS file name with no extension. The usual configuration for a PC uses drives A: and B: for the floppy disk drives, and C: for the hard disk. In this instance, once installed, the first LOCKPC secure area will be accessed as though it were drive D:, the next as drive E: and so on until drive M:.

U5 When installing LOCKPC the user will be asked for information on how much disk space is to be allocated to each secure area. This disk space must be contiguous on the specified drive. It is therefore recommended that before installing LOCKPC one of the commonly available disk compression utilities be run. Once the product is installed, disk fragmentation is no longer of consequence.

U6 In order to ensure the confidentiality of data stored in a disk cache, LOCKPC must be configured and used as described in the User Documentation, reference [LUG].

2.3 Intended Environment

E1 LOCKPC is designed for use on PC compatibles or laptops using any standard DOS operating system software. The LOCKPC software will be installed on the C: drive of the host system from a 3.5" or 5.25" disk.

E2 The PC on which LOCKPC is installed is used in a normal office environment; any physical security measures in place will, of course, enhance the protection provided to the information held on the PC, but is not necessary.

E3 The essence of the system is that both the LOCKPC key and the host PC must be connected in order to access the information or software being protected. When the key is removed, the protected information or software cannot be accessed.

E4 The LOCKPC key will only be allocated on a one key to one user basis, and only to users approved and monitored by the Security Administrator. Once allocated keys must be stored in a lockable secure cabinet by users when they are not in use, as described in reference [LUG].

2.4 Security Objectives

O1 To maintain the confidentiality of a user's data stored on the PC.

2.5 Assumed Threats

T1 A threat exists in that unauthorised users could gain access to information resident on a user's PC. Similarly authorised users may access other users' data without permission. This is countered by the security enforcing functions specified in section 3.

T2 Another possibility is that the LOCKPC device drivers themselves could be tampered with. The device drivers are held on the user's 'logical' drive and as a result cannot be accessed without the LOCKPC key. Therefore this is countered by the security enforcing functions specified in section 3.

T3 It is possible that a user's LOCKPC key may fall into the wrong hands. This is countered by the personnel and physical security measures detailed in the User Documentation, reference [LUG], i.e. environmental requirement E4.

T4 If LOCKPC is used on disk cache systems then information held in the disk cache will be accessible when the key is removed. This threat can be countered by the procedural security measures detailed in the User Documentation, reference [LUG], i.e. method of use requirement U6.

2.6 Summary of Security Features

LOCKPC consists of a software and hardware key combination. The software, which should be installed on the host PC's hard disk, will operate with the hardware key when it is plugged into the parallel printer port. To access protected information, both the key and software must be installed on the PC. By removing the key the information will become inaccessible. The software cannot encrypt/decrypt information if the key is not in place in the parallel port.

Information is encrypted, using the LOCKPC software, when data is written to the LOCKPC secure area. No special commands are needed to write to the secure area; encryption occurs automatically while using straightforward DOS commands. Encrypted information in the secure area may be used in a normal fashion, and the functioning of LOCKPC will be invisible to users.

The objectives of LOCKPC are to prevent unauthorised users from gaining access to data, and to ensure data confidentiality is maintained, by encryption. The security enforcing functions proposed counter the assumed threats T1 and T2, detailed above. The LOCKPC key issued to a user must be present in the parallel printer port before access can be gained to that user's secure areas. The LOCKPC device driver (LOCDRVR.SYS) ensures that data is encrypted when written to the secure area and decrypted when read from that area.

The LOCKPC package does not depend on the operation of any hardware or software other than that specified above.

3 Specification of Security Enforcing Functions

The Security Enforcing Functions provided by LOCKPC are all concerned with access control, and are as follows:

SF1 Only users approved by the security administrator can gain access to their own secure areas, by using the LOCKPC key, which is unique to that individual user.

SF2 Information written to or read from the secure area (or logical drive), is automatically encrypted/decrypted at the time of writing/reading.

4 Definition of Security Mechanisms

The [ITSEC] does not require mechanisms to be specified in the security target except where they are a mandatory requirement to be implemented by the developer. However, it is convenient to identify LOCKPC's mechanisms at this point. This security target therefore provides a single common reference point to be used by all subsequent evaluation documentation.

SM1 LOCKPC uses the device driver (LOCDRVR.SYS) and becomes a 'logical drive'. This logical drive carries out automatic encryption/decryption of data written to or read from the secure area. This mechanism enforces the security enforcing function SF2 specified above.

SM2 At all times the encryption/decryption process uses the hardware key plugged into the parallel printer port. Without this key the information held in the secure areas cannot be decrypted by the software.

SM3 Each LOCKPC device driver may have up to four associated secure areas (or logical drives). If more than four secure areas are required, this may be achieved using one of two simple procedures. Multiple copies of the device driver may be installed by either copying the LOCDRVR.SYS to another name and installing it in the CONFIG.SYS file under that name, or using the same name, but installing it from another directory. In both cases, each installed copy of the device driver may access up to four secure areas.

5 Claimed Minimum Strength of Mechanisms

The minimum strength of mechanism for LOCKPC is *basic*.

6 LOCKPC - Target Level of Assurance

The features provided by LOCKPC make it suitable for evaluation to the ITSEC criteria to level E1 as defined in reference [ITSEC].

7 References

[LUG] LOCKPC User Guide, Version 1.0, 20 August 1992

[ITSEC] Information Technology Security Evaluation Criteria, Version 1.2, June 1991.

Annex B Example Security Target - COMSOL

COMSOL SECURE UNIX SECURITY TARGET

1 Introduction

This document is the security target against which COMSOL Secure UNIX will be evaluated. It provides a specification of the security enforcing functions and a description of the product and the environment in which it is intended for use.

The version of COMSOL that will be evaluated is 3.6.

2 Product Rationale

2.1 Introduction

COMSOL Secure UNIX is an enhancement to the standard AT&T UNIX Release 2 of System 5, in order to provide extra security to a level of assurance suitable for ITSEC level E2. This extra security is provided by the addition of extra functionality in the areas of identification and authentication, accountability and audit, and access control. The product is intended for use on the COMSOL C/386 series of machines.

2.2 Intended Method of Use

The features of COMSOL Secure UNIX are intended to provide secure platforms for use in organisations requiring systems capable of supporting large groups of simultaneous users, using the wide range of standard UNIX office automation utilities available, whilst maintaining the integrity of the information stored.

In addition, COMSOL Secure UNIX provides an operating system for the development of secure systems or applications on which a significant amount of trust can be placed.

More specific guidance on how to operate COMSOL Secure UNIX is provided in the comprehensive user documentation, most specifically the User Guide, reference **[CUG]**, and the Administration Guide, reference **[CSAG]**.

2.3 Intended Environment

COMSOL Secure UNIX is designed to run on the COMSOL C/386 series of machines. This consists of the C80386 main Processor Board which has in itself 16MB of physical memory, SCSI interface for disk and tape drives, ESDI interface for disk drives, a C80387 floating point processor and an Ethernet connectivity option. The C/386 can accept three disk drives, each with a capacity of 800MB. There are no other dependencies on hardware or software.

The C/386 hardware on which the COMSOL Secure UNIX operating system will run should be physically protected and accessible only by authorised staff. All internal and external I/O devices must be SCSI compatible. It is also assumed that operational, administrative, maintenance, security and systems programming staff are trustworthy and appropriately trained, with physical, personnel and security procedures in place to support the security of the system.

2.4 Security Objectives

O1 COMSOL Secure UNIX provides the capability to control and limit access to information stored on the system, using a consistent set of rules. Each user may only control access to the data which he owns.

O2 The system administrator is provided with the means of holding individual users accountable for their actions, where those actions are relevant to security.

2.5 Assumed Threats

The threats detailed in this section have been referenced to the security enforcing functions which are implemented to counter them.

Threat 1

An unauthorised user could attempt to gain access to the privileged accounts by ascertaining the password of that privileged account.

This threat is countered by security enforcing functions F1 and F4 and those personnel and physical procedures designed for the protection of passwords detailed in Chapter 1, reference **[CUG]**. Function F1 ensures that login names and supplied passwords are checked against the password file, /etc/passwd, for validity. All passwords held in this file are encrypted thus preventing the password being ascertained by unauthorised users. Function F4 also ensures that the password file can only be accessed by the privileged **sadmin** user.

Threat 2

An authorised user could attempt to gain access to another user's data.

.........etc.

2.6 Summary of Security Features

COMSOL Secure UNIX will enhance the level of trust for the standard UNIX system through the use of additional functionality. These functions are primarily in the areas of identification and authentication, access control, and accountability and audit.

The security features provided by COMSOL Secure UNIX are those required to satisfy the ITSEC predefined functionality class F-C2. To this end, the product includes mechanisms to implement the identification and authentication of users, and a system of discretionary access control, which enables users to protect objects from unauthorised access. In addition, users of the system are held accountable for their actions by the accounting functions, which ensure that all security relevant actions are recorded, and the auditing tools which filter the amount and type of information gathered. The object reuse mechanism provided prevents processes using memory allocated to other processes and also clears memory addresses freed by processes.

3 Specification of Security Enforcing Functions

COMSOL Secure Unix implements the ITSEC Functionality Class F-C2. The content of this functionality class is replicated below, since it is necessary to identify individual functional requirements for the purposes of subsequent explanation.

The Security Enforcing Functions are specified below, under the F-C2 headings: Identification and Authentication, Access Control, Accountability, Audit and Object Reuse.

3.1 Identification and Authentication

F1 The TOE shall uniquely identify and authenticate users.

F2 Identification and authentication shall take place before all other interaction between the TOE and user.

F3 Other interactions shall only be possible after successful identification and authentication.

F4 Authentication information shall be stored in such a way that is can only be accessed by authorised users.

F5 For every interaction the TOE shall be able to establish the identity of the user.

3.2 Access Control

F6 The TOE shall be able to distinguish and administer access rights between each user and the objects which are subject to the administration of rights, on the basis of an individual user, or on the basis of membership of a group of users, or both.

F7 It shall be possible to completely deny users or user groups access to an object.

F8 It shall be possible to restrict a user's access to an object to those operations which do not modify it.

F9 It shall be possible to grant the access rights to an object down to the granularity of an individual user.

F10 It shall not be possible for anyone who is not an authorised user to grant or revoke access rights to an object.

F11 The administration of rights shall provide controls to limit propagation of access rights.

F12 Only authorised users shall be able to introduce new users or delete or suspend existing users.

F13 For all attempts by users or user groups to access objects which are subject to the administration of rights, the TOE shall verify the validity of the request.

F14 Unauthorised access attempts shall be rejected.

3.3 Accountability

F15 The TOE shall contain an accountability component which is able, for each of the following events, to log that event with the required data:

a) Use of the identification and authentication mechanism, recording date, time, user id, device id, success or failure of access attempt

b) Actions that attempt to exercise access rights to an object which is subject to the administration of rights, recording date, time, user id, object name, type of access attempt, success or failure of access attempt

c) Creation or deletion of an object which is subject to the administration of access rights, recording date, time, user id, name of object, type of action

d) Actions by authorised users affecting the security of the TOE, recording date, time, user id, name of object to which the action relates, type of action (i.e. introduction or suspension of users, introduction or removal of storage media, start up or shutdown of TOE).

F16 Unauthorised users shall not be permitted access to accountability data.

F17 Tools to examine and maintain the accountability files shall exist and be documented.

F18 These tools shall allow the actions of one or more users to be identified selectively.

3.4 Audit

F19 Tools to examine the accountability files for the purpose of audit shall exist and be documented. These tools shall allow the action of one or more users to be identified selectively.

3.5 Object Reuse

F20 All storage objects returned to the TOE shall be treated, before reuse by other subjects, in such a way that no conclusions can be drawn regarding the preceding content.

4 Definition of Security Mechanisms *(optional)*

Although not absolutely necessary under the terms of the ITSEC, this section identifies the various mechanisms provided by COMSOL Secure UNIX. Definition of these mechanisms within the security target (alongside the definition of the security enforcing functions) allows for ease of reference from a single source.

The mechanisms defined in this section are grouped under the same generic headings used for the definition of the security enforcing functions. Each security mechanism has a reference number relating to that of the security enforcing function it implements (e.g. M20 implements F20).

4.1 Identification and Authentication

M1 Whenever a user logs onto the system he is required to enter a login name and password by the **login ( )** process. The supplied information is then checked for validity by this process against the encrypted password entry, corresponding to the user's login name, stored in the file /etc/passwd before access to the system is granted. For details of this process see Section 5.3, reference **[CDD].**

M2 ....etc.

4.2 Access Control

M6 The system administers the standard UNIX access rights, which allows each user to restrict the read, write or execute permissions to objects the user owns. Users can have access permissions restricted by group as well as by user. The System Administrator (sadmin user) is a privileged user who establishes individual access rights, and group memberships when the users are set up on the system and can change access permissions on the files using the **chmod** process. Ordinary users can only change access rights to files they own. For details of the COMSOL access controls see Section 2.3, reference **[CUG]** and Chapter 3, reference **[CDD].**

M7 ....etc.

4.3 Accountability

M15 The system provides the **account** process which produces records of all processes run on the system, recording date, time, user-id, process-id, terminal-id, and descriptor flags indicating the success or failure of the process. The use of the account process is restricted to the privileged **audit** user. For details of the use of this process see Chapter 3, reference **[CSAG]** and Section 4.2, reference **[CDD]**.

M16 ....etc.

4.4 Object Reuse

M20 The Memory Management System of the COMSOL Secure UNIX Operating System allocates memory to processes using the **allocate** process. This operation allocates free space by referring to a lookup table showing the current memory locations allocated to processes. If there is sufficient space in memory to load a process then the **allocate** process calls a **deallocate** process which clears the memory addresses indicated by the lookup table. For details of the memory management routines see Chapter 5, reference **[CDD]**.

5 Claimed Strength of Mechanisms

The minimum strength of mechanisms for COMSOL Secure UNIX is medium.

6 Target Evaluation Level

The target level of assurance for COMSOL Secure UNIX is ITSEC level E2, as defined in reference **[ITSEC]**.

7 References

[CDD] COMSOL Secure UNIX Detailed Design Specification, Version 3.1, June 1989.

[CSAG] COMSOL Secure UNIX System Administration Guide, Version 1.3, July 1990.

[CUG] COMSOL Secure UNIX User Guide, Version 1.1, June 1989.

[ITSEC] Information Technology Security Evaluation Criteria, Version 1.2, June 1991.

INDEX