Common Criteria for Information Technology Security Evaluation

Part 3: Security Assurance Requirements

March 2004

Version 2.4
Revision 256

ASE/APE Trial Use version

CCIMB-2004-03-003

Foreword

This version of the Common Criteria for Information Technology Security Evaluation (CC 2.4) is based on CC v2.2, and includes an updated version of the Protection Profile and Security Target criteria (APE and ASE), together with significant changes in the rest of the CC that were necessary to accommodate these new criteria.

CC version 2.4 consists of the following parts:

– Part 1: Introduction and general model

– Part 2: Security functional requirements

– Part 3: Security assurance requirements

This Legal NOTICE has been placed in all Parts of the CC by request:

The seven governmental organisations (collectively called the "Common Criteria Project Sponsoring Organisations") listed just below, as the joint holders of the copyright in the Common Criteria for Information Technology Security Evaluations, version 2.4 Parts 1 through 3 (called "CC 2.4"), hereby grant non-exclusive license to ISO/IEC to use CC 2.4 in the continued development/maintenance of the ISO/IEC 15408 international standard. However, the Common Criteria Project Sponsoring Organisations retain the right to use, copy, distribute, translate or modify CC 2.4 as they see fit.

Canada: Communications Security Establishment
France: Service Central de la Sécurité des Systèmes d'Information
Germany: Bundesamt für Sicherheit in der Informationstechnik
Netherlands: Netherlands National Communications Security Agency
United Kingdom: Communications-Electronics Security Group
United States: National Institute of Standards and Technology
United States: National Security Agency

Table of Contents

Table of contents

Table of contents

List of figures

List of tables

1 Scope

1 This Part 3 defines the assurance requirements of the CC. It includes the evaluation assurance levels (EALs) that define a scale for measuring assurance, the individual assurance components from which the assurance levels are composed, and the criteria for evaluation of PPs and STs.

1.1 Organisation of CC Part 3

2 Clause 1 is the introduction and paradigm for this CC Part 3.

3 Clause 2 describes the presentation structure of the assurance classes, families, components, and evaluation assurance levels along with their relationships. It also characterises the assurance classes and families found in clauses 8 through 14.

4 Clause 3 provides a brief introduction to the evaluation criteria for PPs, followed by detailed explanations of the families and components that are used for those evaluations.

5 Clause 4 provides detailed definitions of the EALs.

6 Clause 5 provides a brief introduction to the assurance classes and is followed by clauses 6 through 13 that provide detailed definitions of those classes.

7 Annex A provides an explanation of the AVA criteria and examples of their application..

8 Annex B provides a summary of the dependencies between the assurance components.

9 Annex C provides a cross reference between the EALs and the assurance components.

1.2 CC assurance paradigm

10 The purpose of this subclause is to document the philosophy that underpins the CC approach to assurance. An understanding of this subclause will permit the reader to understand the rationale behind the CC Part 3 assurance requirements.

1.2.1 CC philosophy

11 The CC philosophy is that the threats to security and organisational security policy commitments should be clearly articulated and the proposed security measures be demonstrably sufficient for their intended purpose.

12 Furthermore, measures should be adopted that reduce the likelihood of vulnerabilities, the ability to exercise (i.e. intentionally exploit or

unintentionally trigger) a vulnerability, and the extent of the damage that could occur from a vulnerability being exercised. Additionally, measures should be adopted that facilitate the subsequent identification of vulnerabilities and the elimination, mitigation, and/or notification that a vulnerability has been exploited or triggered.

1.2.2 Assurance approach

13 The CC philosophy is to provide assurance based upon an evaluation (active investigation) of the TOE that is to be trusted. Evaluation has been the traditional means of providing assurance and is the basis for prior evaluation criteria documents. In aligning the existing approaches, the CC adopts the same philosophy. The CC proposes measuring the validity of the documentation and of the resulting TOE by expert evaluators with increasing emphasis on scope, depth, and rigour.

14 The CC does not exclude, nor does it comment upon, the relative merits of other means of gaining assurance. Research continues with respect to alternative ways of gaining assurance. As mature alternative approaches emerge from these research activities, they will be considered for inclusion in the CC, which is so structured as to allow their future introduction.

15 Assurance is grounds for confidence that a TOE in its operational environment solves a defined security problem. The CC-approach to gaining assurance is to divide the problem into two subproblems:

a) postulating a set of SFRs for the TOE and gain assurance that a TOE meeting these SFRs in its operational environment will solve the defined security problem;

b) gain assurance that the TOE meets these SFRs.

1.2.2.1 The requirements solve the problem

16 Postulating SFRs and showing that these solve a defined security problem is done in the CC through the use of the Security Target construct.

17 In this Security Target, the security problem is defined, and it is shown how the combination of SFRs and security objectives for the operational environment address this problem.

18 Assurance in the correctness of the Security Target is gained through evaluation (application of the ASE criteria) of the Security Target.

19 More information on Security Targets can be found in CC Part 1 Annex B.

1.2.2.2 The TOE meets the requirements

20 When the ST has been succesfully evaluated, assurance has been established that a TOE meeting the SFRs in the ST, in the operational environment defined in the ST, will solve the security problem that was defined in the ST

21 The next step is gaining assurance that the TOE actually meets these SFRs, and does not contain vulnerabilities. Vulnerabilities can arise through failures in:

a) construction - that is, a TOE does not meet its SFRs and/or vulnerabilities have been introduced as a result of poor constructional standards or incorrect design choices;

b) operation - that is, a TOE has been constructed correctly to correct SFRs but vulnerabilities have been introduced as a result of inadequate controls upon the operation.

22 This assurance is gained through evaluation. Evaluation techniques can include, but are not limited to:

a) analysis and checking of process(es) and procedure(s);

b) checking that process(es) and procedure(s) are being applied;

c) analysis of the correspondence between TOE design representations;

d) analysis of the TOE design representations against the SFRs;

e) verification of proofs;

f) analysis of guidance documents;

g) analysis of functional tests developed and the results provided;

h) independent functional testing;

i) analysis for vulnerabilities (including flaw hypothesis);

j) penetration testing.

1.2.3 The CC evaluation assurance scale

23 The CC philosophy asserts that greater assurance results from the application of greater evaluation effort, and that the goal is to apply the minimum effort required to provide the necessary level of assurance. The increasing level of effort is based upon:

a) depth -- that is, the effort is greater because it is deployed to a finer level of design and implementation detail;

b) rigour -- that is, the effort is greater because it is applied in a more structured, formal manner.

2 Security assurance requirements

2.1 Structures

24 The following subclauses describe the constructs used in representing the assurance classes, families, components, and EALs along with the relationships among them.

25 Figure **1** illustrates the SARs defined in this CC Part 3. Note that the most abstract collection of SARs is referred to as a class. Each class contains assurance families, which then contain assurance components, which in turn contain assurance elements. Classes and families are used to provide a taxonomy for classifying SARs, while components are used to specify SARs in a PP/ST.

2.1.1 Class structure

26 Figure **1** illustrates the assurance class structure.

2.1.1.1 Class name

27 Each assurance class is assigned a unique name. The name indicates the topics covered by the assurance class.

28 A unique short form of the assurance class name is also provided. This is the primary means for referencing the assurance class. The convention adopted is an “A” followed by two letters related to the class name.

2.1.1.2 Class introduction

29 Each assurance class has an introductory subclause that describes the composition of the class and contains supportive text covering the intent of the class.

2.1.1.3 Assurance families

30 Each assurance class contains at least one assurance family. The structure of the assurance families is described in the following subclause.

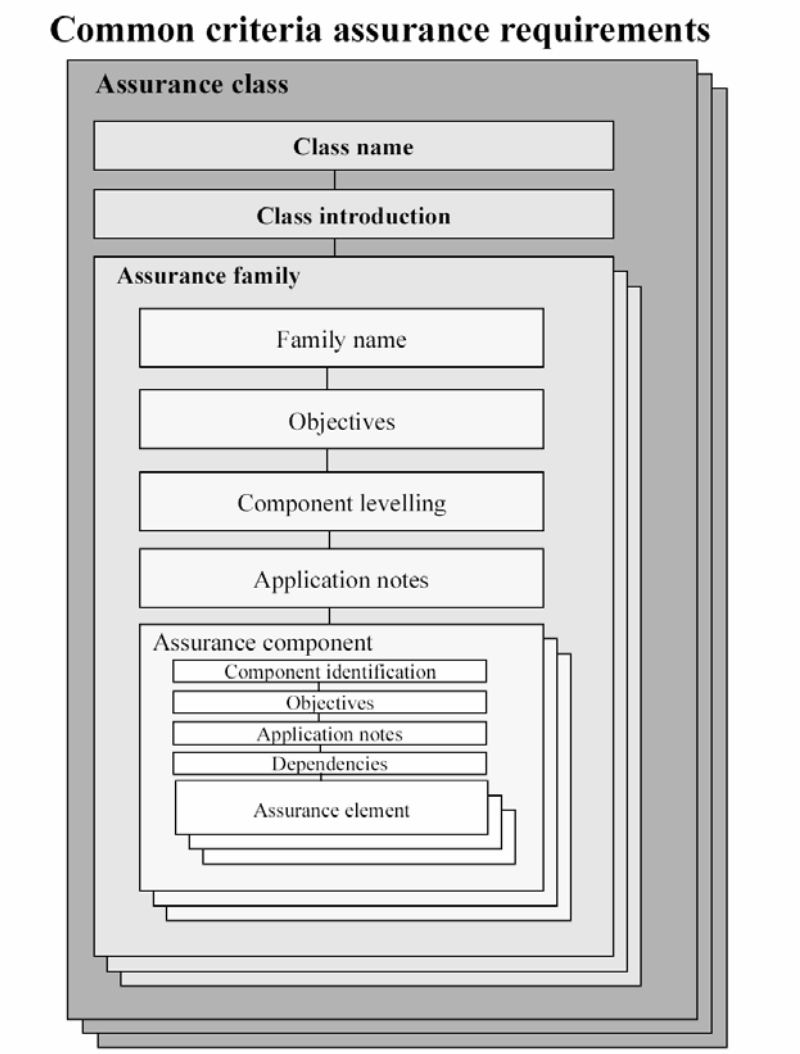

Figure 1 - Assurance class/family/component/element hierarchy

2.1.2 Assurance family structure

31 Figure **1** illustrates the assurance family structure.

2.1.2.1 Family name

32 Every assurance family is assigned a unique name. The name provides descriptive information about the topics covered by the assurance family. Each assurance family is placed within the assurance class that contains other families with the same intent.

33 A unique short form of the assurance family name is also provided. This is the primary means used to reference the assurance family. The convention adopted is that the short form of the class name is used, followed by an underscore, and then three letters related to the family name.

2.1.2.2 Objectives

34 The objectives subclause of the assurance family presents the intent of the assurance family.

35 This subclause describes the objectives, particularly those related to the CC assurance paradigm, that the family is intended to address. The description for the assurance family is kept at a general level. Any specific details required for objectives are incorporated in the particular assurance component.

2.1.2.3 Component levelling

36 Each assurance family contains one or more assurance components. This subclause of the assurance family describes the components available and explains the distinctions between them. Its main purpose is to differentiate between the assurance components once it has been determined that the assurance family is a necessary or useful part of the SARs for a PP/ST.

37 Assurance families containing more than one component are levelled and rationale is provided as to how the components are levelled. This rationale is in terms of scope, depth, and/or rigour.

2.1.2.4 Application notes

38 The application notes subclause of the assurance family, if present, contains additional information for the assurance family. This information should be of particular interest to users of the assurance family (e.g. PP and ST authors, designers of TOEs, evaluators). The presentation is informal and covers, for example, warnings about limitations of use and areas where specific attention may be required.

2.1.2.5 Assurance components

39 Each assurance family has at least one assurance component. The structure of the assurance components is provided in the following subclause.

2.1.3 Assurance component structure

40 Figure **2** illustrates the assurance component structure.

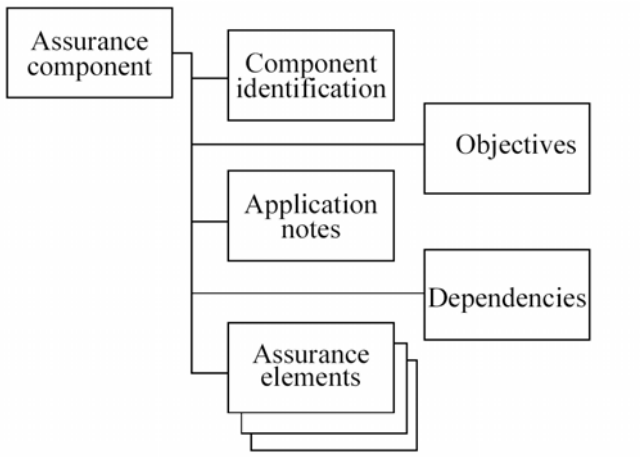

Figure 2 - Assurance component structure

41 The relationship between components within a family is highlighted using a bolding convention. Those parts of the requirements that are new, enhanced or modified beyond the requirements of the previous component within a hierarchy are bolded. The same bolding convention is also used for dependencies.

2.1.3.1 Component identification

42 The component identification subclause provides descriptive information necessary to identify, categorise, register, and reference a component.

43 Every assurance component is assigned a unique name. The name provides descriptive information about the topics covered by the assurance component. Each assurance component is placed within the assurance family that shares its security objective.

44 A unique short form of the assurance component name is also provided. This is the primary means used to reference the assurance component. The convention used is that the short form of the family name is used, followed by a period, and then a numeric character. The numeric characters for the components within each family are assigned sequentially, starting from 1.

2.1.3.2 Objectives

45 The objectives subclause of the assurance component, if present, contains specific objectives for the particular assurance component. For those assurance components that have this subclause, it presents the specific intent of the component and a more detailed explanation of the objectives.

2.1.3.3 Application notes

46 The application notes subclause of an assurance component, if present, contains additional information to facilitate the use of the component.

2.1.3.4 Dependencies

47 Dependencies among assurance components arise when a component is not self-sufficient, and relies upon the presence of another component.

48 Each assurance component provides a complete list of dependencies to other assurance components. Some components may list “No dependencies”, to indicate that no dependencies have been identified. The components depended upon may have dependencies on other components.

49 The dependency list identifies the minimum set of assurance components which are relied upon. Components which are hierarchical to a component in the dependency list may also be used to satisfy the dependency.

50 In specific situations the indicated dependencies might not be applicable. The PP/ST author, by providing rationale for why a given dependency is not applicable, may elect not to satisfy that dependency.

2.1.3.5 Assurance elements

51 A set of assurance elements is provided for each assurance component. An assurance element is a security requirement which, if further divided, would not yield a meaningful evaluation result. It is the smallest security requirement recognised in the CC.

52 Each assurance element is identified as belonging to one of the three sets of assurance elements:

a) Developer action elements: the activities that shall be performed by the developer. This set of actions is further qualified by evidential material referenced in the following set of elements. Requirements for developer actions are identified by appending the letter “D” to the element number.

b) Content and presentation of evidence elements: the evidence required, what the evidence shall demonstrate, and what information the evidence shall convey. Requirements for content and presentation of evidence are identified by appending the letter “C” to the element number.

c) Evaluator action elements: the activities that shall be performed by the evaluator. This set of actions explicitly includes confirmation that the requirements prescribed in the content and presentation of evidence elements have been met. It also includes explicit actions and analysis that shall be performed in addition to that already performed by the developer. Implicit evaluator actions are also to be performed as a result of developer action elements which are not covered by content and presentation of evidence requirements. Requirements for evaluator actions are identified by appending the letter “E” to the element number.

53 The developer actions and content and presentation of evidence define the assurance requirements that are used to represent a developer's responsibilities in demonstrating assurance in the TOE meeting the SFRs of a PP or ST.

54 The evaluator actions define the evaluator's responsibilities in the two aspects of evaluation. The first aspect is validation of the PP/ST, in accordance with the classes APE and ASE in clauses APE: Protection Profile evaluation and ASE: Security Target evaluation. The second aspect is verification of the TOE's conformance with its SFRs and SARs. By demonstrating that the PP/ST is valid and that the requirements are met by the TOE, the evaluator can provide a basis for confidence that the TOE in its operational environment solves the defined security problem.

55 The developer action elements, content and presentation of evidence elements, and explicit evaluator action elements, identify the evaluator effort that shall be expended in verifying the security claims made in the ST of the TOE.

2.1.4 Assurance elements

56 Each element represents a requirement to be met. These statements of requirements are intended to be clear, concise, and unambiguous. Therefore, there are no compound sentences: each separable requirement is stated as an individual element.

2.1.5 EAL structure

57 Figure **3** illustrates the EALs and associated structure defined in this Part 3. Note that while the figure shows the contents of the assurance components, it is intended that this information would be included in an EAL by reference to the actual components defined in the CC.

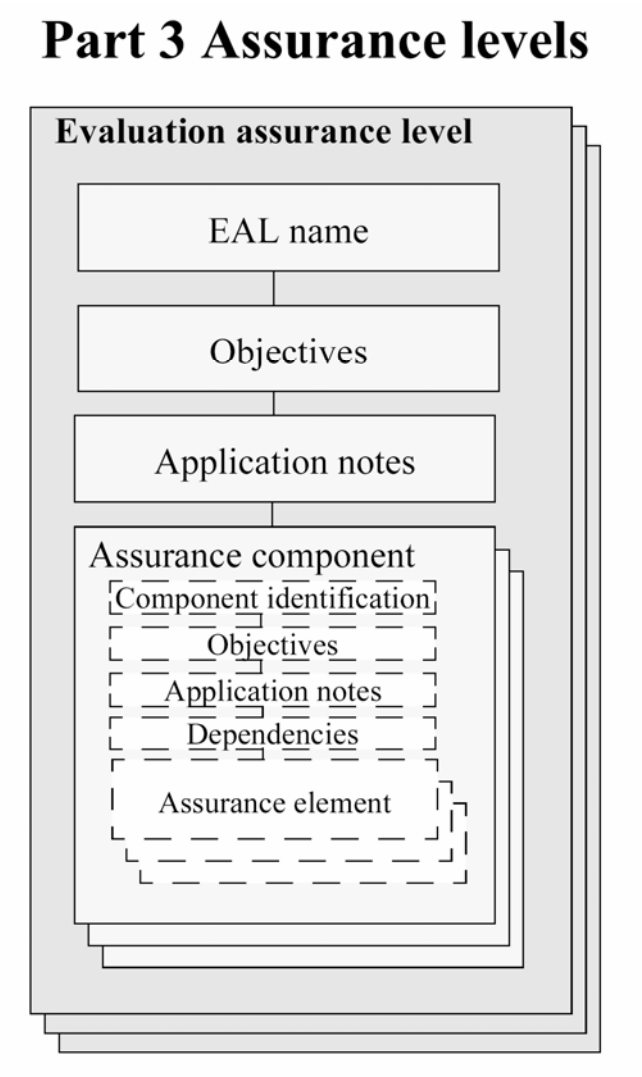

Figure 3 - EAL structure

2.1.5.1 EAL name

58 Each EAL is assigned a unique name. The name provides descriptive information about the intent of the EAL.

59 A unique short form of the EAL name is also provided. This is the primary means used to reference the EAL.

2.1.5.2 Objectives

60 The objectives subclause of the EAL presents the intent of the EAL.

2.1.5.3 Application notes

61 The application notes subclause of the EAL, if present, contains information of particular interest to users of the EAL (e.g. PP and ST authors, designers of TOEs targeting this EAL, evaluators). The presentation is informal and covers, for example, warnings about limitations of use and areas where specific attention may be required.

2.1.5.3.1 Assurance components

62 A set of assurance components have been chosen for each EAL.

63 A higher level of assurance than that provided by a given EAL can be achieved by:

a) including additional assurance components from other assurance families; or

b) replacing an assurance component with a higher level assurance component from the same assurance family.

2.1.5.4 Relationship between assurances and assurance levels

64 Figure **4** illustrates the relationship between the SARs and the assurance levels defined in the CC. While assurance components further decompose into assurance elements, assurance elements cannot be individually referenced by assurance levels. Note that the arrow in the figure represents a reference from an EAL to an assurance component within the class where it is defined.

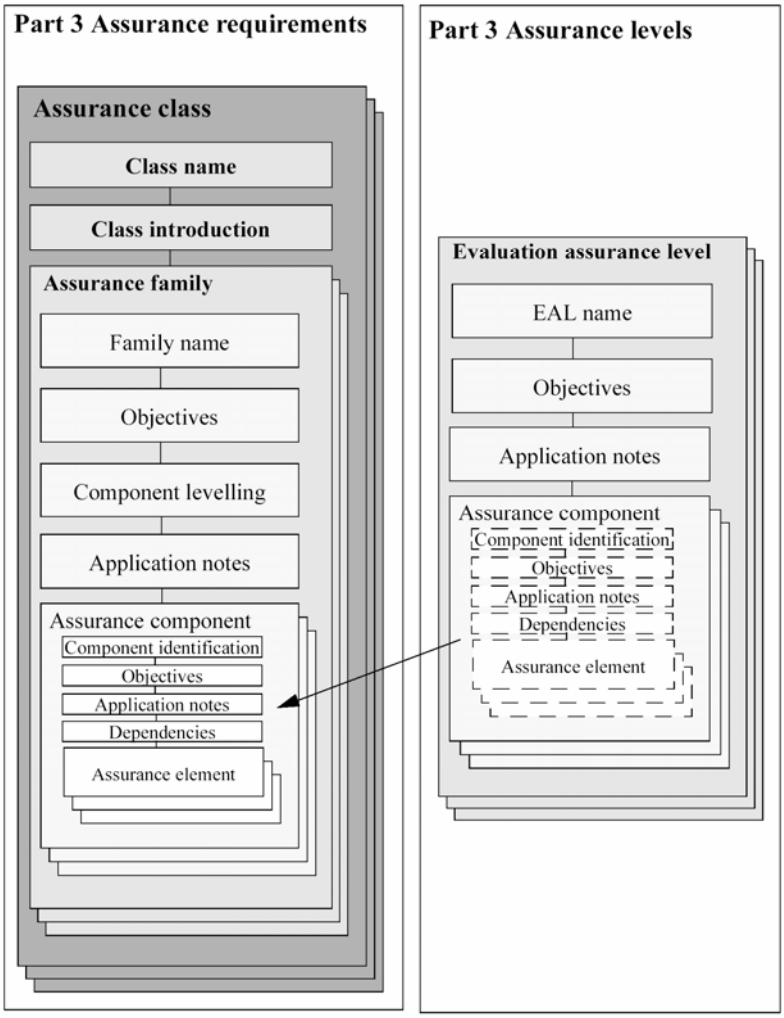

Figure 4 - Assurance and assurance level association

2.2 Component taxonomy

65 This Part 3 contains classes of families and components that are grouped on the basis of related assurance. At the start of each class is a diagram that indicates the families in the class and the components in each family.

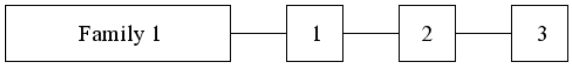

Figure 5 - Sample class decomposition diagram

66 In Figure **5**, above, the class as shown contains a single family. The family contains three components that are linearly hierarchical (i.e. component 2 requires more than component 1, in terms of specific actions, specific evidence, or rigour of the actions or evidence). The assurance families in this Part 3 are all linearly hierarchical, although linearity is not a mandatory criterion for assurance families that may be added in the future.

2.3 Usage of terms in Part 3

67 The following is a list of terms which are used in a precise way in this Part 3. They do not merit inclusion in the glossary because they are general English terms and their usage, though restricted to the explanations given below, is in conformance with dictionary definitions. However, those explanations of the terms were used as guidance in the development of this Part 3 and should be helpful for general understanding.

68 Coherent :

An entity is logically ordered and has a discernible meaning. For documentation, this addresses both the actual text and the structure of the document, in terms of whether it is understandable by its target audience.

69 Complete :

All necessary parts of an entity have been provided. In terms of documentation, this means that all relevant information is covered in the documentation, at such a level of detail that no further explanation is required at that level of abstraction.

70 Confirm :

This term is used to indicate that something needs to be reviewed in detail, and that an independent determination of sufficiency needs to be made. The level of rigour required depends on the nature of the subject matter. This term is only applied to evaluator actions.

71 Consistent :

This term describes a relationship between two or more entities, indicating that there are no apparent contradictions between these entitieAssurance categorisations.

72 Counter (verb) :

This term is typically used in the context that the impact of a particular threat is mitigated but not necessarily eradicated.

73 Demonstrate :

This term refers to an analysis leading to a conclusion, which is less rigourous than a “proof”.

74 Describe :

This term requires that certain, specific details of an entity be provided.

75 Determine :

This term requires an independent analysis to be made, with the objective of reaching a particular conclusion. The usage of this term differs from “confirm” or “verify”, since these other terms imply that an analysis has already been performed which needs to be reviewed, whereas the usage of “determine” implies a truly independent analysis, usually in the absence of any previous analysis having been performed.

76 Ensure :

This term, used by itself, implies a strong causal relationship between an action and its consequences. This term is typically preceded by the word “helps”, which indicates that the consequence is not fully certain, on the basis of that action alone.

77 Exhaustive :

This term is used in the CC with respect to conducting an analysis or other activity. It is reAssurance categorisationlated to “systematic” but is considerably stronger, in that it indicates not only that a methodical approach has been taken to perform the analysis or activity according to an unambiguous plan, but that the plan that was followed is sufficient to ensure that all possible avenues have been exercised.

78 Explain :

This term differs from both “describe” and “demonstrate”. It is intended to answer the question “Why?” without actually attempting to argue that the course of action that was taken was necessarily optimal.

79 Internally consistent :

There are no apparent contradictions between any aspects of an entity. In terms of documentation, this means that there can be no statements within the documentation that can be taken to contradict each other.

80 Justification :

This term refers to an analysis leading to a conclusion, but is more rigorous than a demonstration. This term requires significant rigour in terms of very carefully and thoroughly explaining every step of a logical argument.

81 Prove :

This refers to a formal analysis in its mathematical sense. It is completely rigourous in all ways. Typically, “prove” is used when there is a desire to show correspondence between two TSF representations at a high level of rigour.

82 Specify :

This term is used in the same context as “describe”, but is intended to be more rigourous and precise. It is very similar to “define”.

83 Trace (verb) :

This term is used to indicate that an informal correspondence is required between two entities with only a minimal level of rigour.

84 Verify :

This term is similar in context to “confirm”, but has more rigourous connotations. This term when used in the context of evaluator actions indicates that an independent effort is required of the evaluator.

2.4 Assurance categorisation

85 The assurance classes, families, and the abbreviation for each family are shown in Table 1 Assurance family breakdown and mapping.

Assurance Class	Assurance Family	Abbreviated Name
ACM: Configuration management	CM automation (ACM_AUT)	ACM_AUT
	CM capabilities (ACM_CAP)	ACM_CAP
	CM scope (ACM_SCP)	ACM_SCP
ADO: Delivery and operation	Delivery (ADO_DEL)	ADO_DEL
	Installation, generation and start-up (ADO_IGS)	ADO_IGS
ADV: Development	Functional specification (ADV_FSP)	ADV_FSP
	High-level design (ADV_HLD)	ADV_HLD
	Implementation representation (ADV_IMP)	ADV_IMP
	TSF internals (ADV_INT)	ADV_INT
	Low-level design (ADV_LLD)	ADV_LLD
	Representation correspondence (ADV_RCR)	ADV_RCR
	Security policy modeling (ADV_SPM)	ADV_SPM
AGD: Guidance documents	Administrator guidance (AGD_ADM)	AGD_ADM
	User guidance (AGD_USR)	AGD_USR
ALC: Life cycle support	Development security (ALC_DVS)	ALC_DVS
	Flaw remediation (ALC_FLR)	ALC_FLR
	Life cycle definition	ALC_LCD

Assurance Class	Assurance Family	Abbreviated Name
	(ALC_LCD)	
	Tools and techniques (ALC_TAT)	ALC_TAT
ASE: Security Target evaluation	Conformance claims (ASE_CCL)	ASE_CCL
	Extended components definition (ASE_ECD)	ASE_ECD
	ST introduction (ASE_INT)	ASE_INT
	Security objectives (ASE_OBJ)	ASE_OBJ
	Security requirements (ASE_REQ)	ASE_REQ
	Security problem definition (ASE_SPD)	ASE_SPD
	TOE summary specification (ASE_TSS)	ASE_TSS
ATE: Tests	Coverage (ATE_COV)	ATE_COV
	Depth (ATE_DPT)	ATE_DPT
	Functional tests (ATE_FUN)	ATE_FUN
	Independent testing (ATE_IND)	ATE_IND
AVA: Vulnerability assessment	Covert channel analysis (AVA_CCA)	AVA_CCA
	Misuse (AVA_MSU)	AVA_MSU
	Vulnerability analysis (AVA_VLA)	AVA_VLA

Table 1 Assurance family breakdown and mapping

2.5 Assurance class and family overview

86 The following summarises the assurance classes and families of clauses 6-13. These classes and family summaries are presented in the same order as they appear in clauses 6-13.

2.5.1 Class ACM:Configuration management

87 Configuration management (CM) helps to ensure that the integrity of the TOE is preserved, by requiring discipline and control in the processes of refinement and modification of the TOE and other related information. CM prevents unauthorised modifications, additions, or deletions to the TOE, thus providing assurance that the TOE and documentation used for evaluation are the ones prepared for distribution.

2.5.1.1 CM automation (ACM_AUT)

88 Configuration management automation establishes the level of automation used to control the configuration items.

2.5.1.2 CM capabilities (ACM_CAP)

89 Configuration management capabilities define the characteristics of the configuration management system.

2.5.1.3 CM scope (ACM_SCP)

90 Configuration management scope indicates the TOE items that need to be controlled by the configuration management system.

2.5.2 Class ADO:Delivery and operation

91 Assurance class ADO: Delivery and operation defines requirements for the measures, procedures, and standards concerned with secure delivery, installation, and operational use of the TOE, ensuring that the security protection offered by the TOE is not compromised during transfer, installation, start-up, and operation.

2.5.2.1 Delivery (ADO_DEL)

92 Delivery covers the procedures used to maintain security during transfer of the TOE to the user, both on initial delivery and as part of subsequent modification. It includes special procedures or operations required to demonstrate the authenticity of the delivered TOE. Such procedures and measures are the basis for ensuring that the security protection offered by the TOE is not compromised during transfer. While compliance with the delivery requirements cannot always be determined when a TOE is evaluated, it is possible to evaluate the procedures that a developer has developed to distribute the TOE to users.

2.5.2.2 Installation, generation and start-up (ADO_IGS)

93 Installation, generation, and start-up requires that the copy of the TOE is configured and activated by the administrator to exhibit the same protection properties as the master copy of the TOE. The installation, generation, and start-up procedures provide confidence that the administrator will be aware of the TOE configuration parameters and how they can affect the TSF.

2.5.3 Class ADV:Development

94 Assurance class ADV: Development defines requirements for the stepwise refinement of the TSF from the TOE summary specification in the ST down to the actual implementation. Each of the resulting TSF representations provide information to help the evaluator determine whether the TOE meets its SFRs.

2.5.3.1 Functional specification (ADV_FSP)

95 The functional specification describes the TSF, and must be a complete and accurate instantiation of the SFRs. The functional specification also details the external interface to the TOE. Users of the TOE are expected to interact with the TSF through this interface.

2.5.3.2 High-level design (ADV_HLD)

96 The high-level design is a top level design specification that refines the TSF functional specification into the major constituent parts of the TSF. The high level design identifies the basic structure of the TSF and the major hardware, firmware, and software elements.

2.5.3.3 Implementation representation (ADV_IMP)

97 The implementation representation is the least abstract representation of the TSF. It captures the detailed internal workings of the TSF in terms of source code, hardware drawings, etc., as applicable.

2.5.3.4 TSF internals (ADV_INT)

98 The TSF internals requirements specify the requisite internal structuring of the TSF.

2.5.3.5 Low-level design (ADV_LLD)

99 The low-level design is a detailed design specification that refines the high-level design into a level of detail that can be used as a basis for programming and/or hardware construction.

2.5.3.6 Representation correspondence (ADV_RCR)

100 The representation correspondence is a demonstration of mappings between all adjacent pairs of available TSF representations, from the functional specification through to the least abstract TSF representation that is provided.

2.5.3.7 Security policy modeling (ADV_SPM)

101 Security policy models are structured representations of security policies of the TSP, and are used to provide increased assurance that the functional specification corresponds to the security policies of the TSP, and ultimately to the SFRs. This is achieved via correspondence mappings between the functional specification, the security policy model, and the security policies that are modelled.

2.5.4 Class AGD:Guidance documents

102 Assurance class AGD: Guidance documents defines requirements directed at the understandability, coverage and completeness of the operational documentation provided by the developer. This documentation, which provides two categories of information, for users and for administrators, is an important factor in the secure operation of the TOE.

2.5.4.1 Administrator guidance (AGD_ADM)

103 Requirements for administrative guidance help ensure that the environmental constraints can be understood by administrators and operators of the TOE. Administrative guidance is the primary means available to the developer for

providing the TOE administrators with detailed, accurate information of how to administer the TOE in a secure manner and how to make effective use of the TSF privileges and protection functions.

2.5.4.2 User guidance (AGD_USR)

104 Requirements for user guidance help ensure that users are able to operate the TOE in a secure manner (e.g. the usage constraints assumed by the PP or ST must be clearly explained and illustrated). User guidance is the primary vehicle available to the developer for providing the TOE users with the necessary background and specific information on how to correctly use the TOE's protection functions. User guidance must do two things. First, it needs to explain what the user-visible security functions do and how they are to be used, so that users are able to consistently and effectively protect their information. Second, it needs to explain the user's role in maintaining the TOE's security.

2.5.5 Class ALC:Life cycle support

105 Assurance class ALC: Life cycle support defines requirements for assurance through the adoption of a well defined life-cycle model for all the steps of the TOE development, including flaw remediation procedures and policies, correct use of tools and techniques and the security measures used to protect the development environment.

2.5.5.1 Development security (ALC_DVS)

106 Development security covers the physical, procedural, personnel, and other security measures used in the development environment. It includes physical security of the development location(s) and controls on the selection and hiring of development staff.

2.5.5.2 Flaw remediation (ALC_FLR)

107 Flaw remediation ensures that flaws discovered by the TOE consumers will be tracked and corrected while the TOE is supported by the developer. While future compliance with the flaw remediation requirements cannot be determined when a TOE is evaluated, it is possible to evaluate the procedures and policies that a developer has in place to track and repair flaws, and to distribute the repairs to consumers.

2.5.5.3 Life cycle definition (ALC_LCD)

108 Life cycle definition establishes that the engineering practices used by a developer to produce the TOE include the considerations and activities identified in the development process and operational support requirements. Confidence in the correspondence between the requirements and the TOE is greater when security analysis and the production of evidence are done on a regular basis as an integral part of the development process and operational support activities. It is not the intent of this component to dictate any specific development process.

2.5.5.4 Tools and techniques (ALC_TAT)

109 Tools and techniques addresses the need to define the development tools being used to analyse and implement the TOE. It includes requirements concerning the development tools and implementation dependent options of those tools.

2.5.6 Class ASE:Security Target evaluation

110 Assurance class ASE: Security Target evaluation defines requirements for the evaluation of an ST, to demonstrate that the ST is sound and internally consistent, and, if the ST is based on one or more PPs or packages, that the ST is a correct instantiation of these PPs and packages.

2.5.6.1 Conformance claims (ASE_CCL)

111 Conformance claims describes how the Security Target conforms to Parts 2 and Part 3 of the CC, to Protection Profiles and to packages.

2.5.6.2 Extended components definition (ASE_ECD)

112 Extended components are defined wherever it is impossible to clearly express requirements using only components from CC Part 2 and/or CC Part 3.

2.5.6.3 ST introduction (ASE_INT)

113 The ST introduction describes the TOE in a narrative way on three levels of abstraction.

2.5.6.4 Security objectives (ASE_OBJ)

114 Security objectives are a concise statement of the intended response to the security problem defined in the Security problem definition (ASE_SPD) family.

2.5.6.5 Security requirements (ASE_REQ)

115 The SFRs form a clear, unambiguous and canonical description of the expected security behavior of the TOE. The SARs form a clear, unambiguous and canonical description of the expected activities that will be undertaken to gain assurance in the TOE.

2.5.6.6 Security problem definition (ASE_SPD)

116 The security problem definition defines the problem addressed by the TOE, the operational environment of the TOE and the development environment of the TOE.

2.5.6.7 TOE summary specification (ASE_TSS)

117 The TOE Summary specification allows evaluators and potential consumers of the TOE to understand how the TOE meets its SFRs.

2.5.7 Class ATE:Tests

118 Assurance class ATE: Tests states testing requirements that demonstrate that the TOE matches its design descriptions as provided in the ADV: Development class.

2.5.7.1 Coverage (ATE_COV)

119 Coverage deals with the completeness of the functional tests performed by the developer on the TOE. It addresses the extent to which the TSF is tested.

2.5.7.2 Depth (ATE_DPT)

120 Depth deals with the level of detail to which the developer tests the TSF. Testing of is based upon increasing depth of information derived from analysis of the TSF representations.

2.5.7.3 Functional tests (ATE_FUN)

121 Functional testing establishes that the tests performed by the developer are performed and documented correctly.

2.5.7.4 Independent testing (ATE_IND)

122 Independent testing specifies the degree to which the testing of the TSF must be performed by a party other than the developer (e.g. a third party). This family adds value by the introduction of tests that are not part of the developers tests.

2.5.8 Class AVA:Vulnerability assessment

123 Assurance class AVA: Vulnerability assessment defines requirements directed at the identification of exploitable vulnerabilities. Specifically, it addresses those vulnerabilities introduced in the construction, operation, misuse, or incorrect configuration of the TOE.

2.5.8.1 Covert channel analysis (AVA_CCA)

124 Covert channel analysis is directed towards the discovery and analysis of unintended communications channels that can be exploited to violate the TSP.

2.5.8.2 Misuse (AVA_MSU)

125 Misuse analysis investigates whether an administrator or user, with an understanding of the guidance documentation, would reasonably be able to

determine if the TOE is configured and operating in a manner that is insecure.

2.5.8.3 Vulnerability analysis (AVA_VLA)

126 Vulnerability analysis consists of the identification of vulnerabilities potentially introduced in the different refinement steps of the development. These potential vulnerabilities are assessed through penetration testing to determine whether they could, in practice, be exploitable to compromise the TSP.

3 Class APE: Protection Profile evaluation

127 Evaluating a PP is required to demonstrate that the PP is sound and internally consistent, and, if the PP is based on one or more other other PPs or on packages, that the PP is a correct instantiation of these PPs and packages. These properties are necessary for the PP to be suitable for use as the basis for writing an ST.

128 Figure **6** shows the families within this class, and the hierarchy of components within the families.

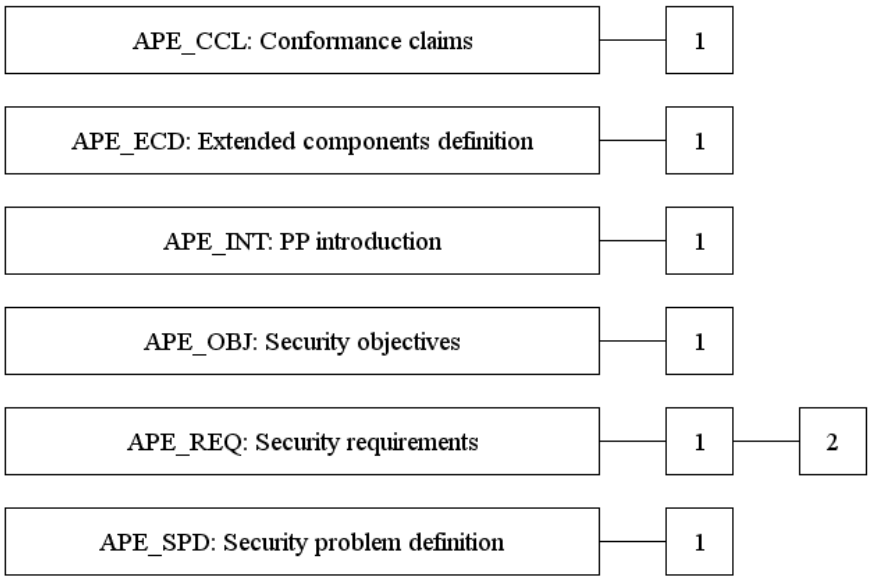

Figure 6 - APE: Protection Profile evaluation class decomposition

3.1 Conformance claims (APE_CCL)

Objectives

129 The objective of this family is to determine the validity of the conformance claim. In addition, this family specifies how STs are to claim conformance with the PP.

APE_CCL.1 Conformance claims

Dependencies

APE_INT.1 PP introduction
ASE_ECD.1 Extended components definition
ASE_REQ.1 Stated security requirements

Developer action elements

APE_CCL.1.1D The developer shall provide a conformance claim.

APE_CCL.1.2D The developer shall provide a conformance claim rationale.

APE_CCL.1.3D The developer shall provide a conformance statement.

Content and presentation of evidence elements

APE_CCL.1.1C The conformance claim shall contain a CC conformance claim that identifies the version of the CC to which the PP claims conformance.

APE_CCL.1.2C The CC conformance claim shall describe the conformance of the PP to CC Part 2 as either CC Part 2 conformant or CC Part 2 extended.

APE_CCL.1.3C The CC conformance claim shall describe the conformance of the PP to CC Part 3 as either CC Part 3 conformant or CC Part 3 extended.

APE_CCL.1.4C The CC conformance claim shall be consistent with the extended components definition.

APE_CCL.1.5C The conformance claim shall identify all PPs and security requirement packages to which the PP claims conformance.

APE_CCL.1.6C The conformance claim shall describe any conformance of the PP to a package as either package-conformant or package-augmented.

APE_CCL.1.7C The conformance claims rationale shall demonstrate that the TOE type is consistent with the TOE type in the PPs for which conformance is being claimed.

APE_CCL.1.8C The conformance claims rationale shall demonstrate that the statement of the security problem definition is consistent with the statement of the security problem definition in the PPs for which conformance is being claimed.

APE_CCL.1.9C The conformance claims rationale shall demonstrate that the statement of objectives is consistent with the statement of objectives in the PPs for which conformance is being claimed.

APE_CCL.1.10C The conformance claims rationale shall demonstrate that the statement of security requirements is consistent with the statement of security requirements in the PPs for which conformance is being claimed.

APE_CCL.1.11C The conformance claims rationale shall demonstrate that all operations of the security requirements that were taken from a PP are completed consistently with the respective PP.

APE_CCL.1.12C The conformance claims rationale shall demonstrate that the statement of security requirements is consistent with the statement of security requirements in the security requirements package for which conformance is being claimed.

APE_CCL.1.13C The conformance claims rationale shall demonstrate that all operations of the security requirements in the PP that were taken from a package are completed consistently with the respective security requirement package.

APE_CCL.1.14C The conformance statement shall describe the conformance required of any PPs/STs as exact-PP, strict-PP or demonstrable-PP -conformance for the PP.

Evaluator action elements

APE_CCL.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

3.2 Extended components definition (APE_ECD)

APE_ECD.1 Extended components definition

Developer action elements

APE_ECD.1.1D The PP developer shall provide a statement of security requirements

APE_ECD.1.2D The PP developer shall provide an extended components definition.

Content and presentation of evidence elements

APE_ECD.1.1C The statement of security requirements shall identify all extended security requirements.

APE_ECD.1.2C The extended components definition shall define an extended component for each extended security requirement.

APE_ECD.1.3C The extended components definition shall describe how each extended component is related to the existing CC components, families, and classes.

APE_ECD.1.4C The extended components definition shall use the existing CC components, families, classes, and methodology as a model for presentation.

APE_ECD.1.5C The extended components shall consist of measurable and objective elements such that compliance or noncompliance to these elements can be demonstrated.

Evaluator action elements

APE_ECD.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

APE_ECD.1.2E The evaluator shall confirm that no extended component can be clearly expressed using existing components.

3.3 PP introduction (APE_INT)

Objectives

130 The objective of this family is to describe the TOE in a narrative way.

131 Evaluation of the PP introduction is required to demonstrate that the PP is correctly identified, and that the PP reference and TOE overview are consistent with each other.

APE_INT.1 PP introduction

Developer action elements

APE_INT.1.1D The PP developer shall provide a PP introduction.

Content and presentation of evidence elements

APE_INT.1.1C The PP introduction shall contain a PP reference and a TOE overview.

APE_INT.1.2C The PP reference shall uniquely identify the PP.

APE_INT.1.3C The TOE overview shall summarise the usage and major security features of the TOE.

APE_INT.1.4C The TOE overview shall identify the TOE type.

APE_INT.1.5C The TOE overview shall identify any non-TOE hardware/software/firmware available to the TOE.

Evaluator action elements

APE_INT.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

3.4 Security objectives (APE_OBJ)

Objectives

132 The security objectives are a concise statement of the intended response to the security problem defined through the Security problem definition (APE_SPD) family.

133 Evaluation of the security objectives is required to demonstrate that the security objectives adequately and completely address the security problem definition, that the division of this problem between the TOE, its development environment, and its operational environment is clearly defined, and that the security objectives are internally consistent.

APE_OBJ.1 Security objectives

Dependencies

APE_SPD.1 Security problem definition

Developer action elements

APE_OBJ.1.1D The PP developer shall provide a statement of security objectives.

APE_OBJ.1.2D The PP developer shall provide a security objectives rationale.

Content and presentation of evidence elements

APE_OBJ.1.1C The statement of security objectives shall describe the security objectives for the TOE.

APE_OBJ.1.2C The security objectives rationale shall trace each security objective for the TOE back to threats countered by that security objective and OSPs met by that security objective.

APE_OBJ.1.3C The statement of security objectives shall describe the security objectives for the development environment.

APE_OBJ.1.4C The security objectives rationale shall trace each security objective for the development environment back to threats countered by that security objective and OSPs met by that security objective.

APE_OBJ.1.5C The statement of security objectives shall describe the security objectives for the operational environment

APE_OBJ.1.6C The security objectives rationale shall trace each security objective for the operational environment back to threats countered by that security objective, OSPs enforced by that security objective, and assumptions upheld by that security objective.

APE_OBJ.1.7C The security objectives rationale shall demonstrate that the security objectives counter all threats.

APE_OBJ.1.8C The security objectives rationale shall demonstrate that the security objectives enforce all OSPs.

APE_OBJ.1.9C The security objectives rationale shall demonstrate that the security objectives for the operational environment uphold all assumptions.

Evaluator action elements

APE_OBJ.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

APE_OBJ.1.2E The evaluator shall confirm that the statement of security objectives is internally consistent.

3.5 Security requirements (APE_REQ)

Objectives

134 The SFRs form a clear, unambiguous and canonical description of the expected security behaviour of the TOE. The SARs form a clear,

unambiguous and canonical description of the expected activities that will be undertaken to gain assurance in the TOE.

135 Evaluation of the security requirements is required to ensure that they are clear, unambiguous and canonical.

Component levelling

136 The components in this family are levelled on whether they are stated as is, or whether they are derived from security objectives for the TOE and security objectives for the development environment.

APE_REQ.1 Stated security requirements

Dependencies

APE_ECD.1 Extended components definition

Content and presentation of evidence elements

APE_REQ.1.1C The statement of security requirements shall describe the SFRs and the SARs.

APE_REQ.1.2C The statement of security requirements shall identify all operations on the security requirements.

APE_REQ.1.3C All operations shall be performed correctly.

Evaluator action elements

APE_REQ.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

APE_REQ.1.2E The evaluator shall confirm that the statement of security requirements is internally consistent.

APE_REQ.2 Derived security requirements

Dependencies

APE_OBJ.1 Security objectives
APE_ECD.1 Extended components definition

Developer action elements

APE_REQ.2.1D The developer shall provide a security requirements rationale.

Content and presentation of evidence elements

APE_REQ.2.1C The statement of security requirements shall describe the SFRs and the SARs.

APE_REQ.2.2C The statement of security requirements shall identify all operations on the security requirements.

APE_REQ.2.3C All operations shall be performed correctly.

APE_REQ.2.4C Each dependency of the security requirements shall either be satisfied, or the security requirements rationale shall justify the dependency not being satisfied.

APE_REQ.2.5C The security requirements rationale shall trace each SFR back to the security objectives for the TOE.

APE_REQ.2.6C The security requirements rationale shall demonstrate that the SFRs meet all security objectives for the TOE.

APE_REQ.2.7C The security requirements rationale shall trace each SAR back to the security objectives for the development environment.

APE_REQ.2.8C The security requirements rationale shall demonstrate that the SARs meet all security objectives for the development environment.

Evaluator action elements

APE_REQ.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

APE_REQ.2.2E The evaluator shall confirm that the statement of security requirements is internally consistent.

3.6 Security problem definition (APE_SPD)

Objectives

137 This part of the PP defines the security problem to be addressed by the TOE, the operational environment of the TOE, and the development environment of the TOE.

138 Evaluation of the security problem definition is required to demonstrate that the security problem intended to be addressed by the TOE, its operational environment, and its development environment, is clearly defined.

APE_SPD.1 Security problem definition

Developer action elements

APE_SPD.1.1D The PP developer shall provide a security problem definition.

Content and presentation of evidence elements

APE_SPD.1.1C The security problem definition shall describe the threats.

APE_SPD.1.2C All threats shall be described in terms of a threat agent, an asset, and an adverse action.

APE_SPD.1.3C The security problem definition shall describe the OSPs.

APE_SPD.1.4C The security problem definition shall describe the assumptions about the operational environment of the TOE.

Evaluator action elements

APE_SPD.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

4 Evaluation assurance levels

139 The Evaluation Assurance Levels (EALs) provide an increasing scale that balances the level of assurance obtained with the cost and feasibility of acquiring that degree of assurance. The CC approach identifies the separate concepts of assurance in a TOE at the end of the evaluation, and of maintenance of that assurance during the operational use of the TOE.

140 It is important to note that not all families and components from CC Part 3 are included in the EALs. This is not to say that these do not provide meaningful and desirable assurances. Instead, it is expected that these families and components will be considered for augmentation of an EAL in those PPs and STs for which they provide utility.

4.1 Evaluation assurance level (EAL) overview

141 Table 2 Evaluation assurance level summary represents a summary of the EALs. The columns represent a hierarchically ordered set of EALs, while the rows represent assurance families. Each number in the resulting matrix identifies a specific assurance component where applicable.

142 As outlined in the next subclause, seven hierarchically ordered evaluation assurance levels are defined in the CC for the rating of a TOE's assurance. They are hierarchically ordered inasmuch as each EAL represents more assurance than all lower EALs. The increase in assurance from EAL to EAL is accomplished by substitution of a hierarchically higher assurance component from the same assurance family (i.e. increasing rigour, scope, and/or depth) and from the addition of assurance components from other assurance families (i.e. adding new requirements).

143 These EALs consist of an appropriate combination of assurance components as described in clause 2 of this Part 3. More precisely, each EAL includes no more than one component of each assurance family and all assurance dependencies of every component are addressed.

144 While the EALs are defined in the CC, it is possible to represent other combinations of assurance. Specifically, the notion of “augmentation” allows the addition of assurance components (from assurance families not already included in the EAL) or the substitution of assurance components (with another hierarchically higher assurance component in the same assurance family) to an EAL. Of the assurance constructs defined in the CC, only EALs may be augmented. The notion of an “EAL minus a constituent assurance component” is not recognised by the standard as a valid claim. Augmentation carries with it the obligation on the part of the claimant to justify the utility and added value of the added assurance component to the EAL. An EAL may also be augmented with extended assurance requirements.

Assurance class	Assurance Family	Assurance Components by Evaluation Assurance Level						
		EAL1	EAL2	EAL3	EAL4	EAL5	EAL6	EAL7
Configuration management	ACM_AUT				**1**	1	**2**	2
	ACM_CAP	**1**	**2**	**3**	**4**	4	**5**	5
	ACM_SCP			**1**	**2**	**3**	3	3
Delivery and operation	ADO_DEL		**1**	1	**2**	2	2	**3**
	ADO_IGS	**1**	1	1	1	1	1	1
Development	ADV_FSP	**1**	1	1	**2**	**3**	3	**4**
	ADV_HLD		**1**	**2**	2	**3**	**4**	**5**
	ADV_IMP				**1**	**2**	**3**	3
	ADV_INT					**1**	**2**	**3**
	ADV_LLD				**1**	1	**2**	2
	ADV_RCR	**1**	1	1	1	**2**	2	**3**
	ADV_SPM				**1**	**3**	3	3
Guidance documents	AGD_ADM	**1**	1	1	1	1	1	1
	AGD_USR	**1**	1	1	1	1	1	1
Life cycle support	ALC_DVS			**1**	1	1	**2**	2
	ALC_FLR							
	ALC_LCD				**1**	**2**	2	**3**
	ALC_TAT				**1**	**2**	**3**	3
Security Target evaluation	ASE_CCL	**1**	1	1	1	1	1	1
	ASE_ECD	**1**	1	1	1	1	1	1
	ASE_INT	**1**	1	1	1	1	1	1
	ASE_OBJ		**1**	1	1	1	1	1
	ASE_REQ	**1**	**2**	2	2	2	2	2
	ASE_SPD		**1**	1	1	1	1	1
	ASE_TSS	**1**	1	1	1	1	1	1
Tests	ATE_COV		**1**	**2**	2	2	**3**	3
	ATE_DPT			**1**	1	**2**	2	**3**
	ATE_FUN		**1**	1	1	1	**2**	2
	ATE_IND	**1**	**2**	2	2	2	2	**3**
Vulnerability assessment	AVA_CCA					**1**	**2**	2
	AVA_MSU			**1**	**2**	2	**3**	3
	AVA_VLA		**1**	1	**2**	**3**	**4**	4

Table 2 Evaluation assurance level summary

4.2 Evaluation assurance level details

145 The following subclauses provide definitions of the EALs, highlighting differences between the specific requirements and the prose characterisations of those requirements using bold type.

4.3 Evaluation assurance level 1 (EAL1) - functionally tested

Objectives

146 EAL1 is applicable where some confidence in correct operation is required, but the threats to security are not viewed as serious. It will be of value where independent assurance is required to support the contention that due care has been exercised with respect to the protection of personal or similar information.

147 EAL1 requires only a limited security target. It is sufficient to simply state the SFRs that the TOE must meet, rather than deriving them from threats, OSPs and assumptions through security objectives.

148 EAL1 provides an evaluation of the TOE as made available to the customer, including independent testing against a specification, and an examination of the guidance documentation provided. It is intended that an EAL1 evaluation could be successfully conducted without assistance from the developer of the TOE, and for minimal outlay.

149 An evaluation at this level should provide evidence that the TOE functions in a manner consistent with its documentation, and that it provides useful protection against identified threats.

Assurance components

150 EAL1 provides a basic level of assurance by a limited security target and an analysis of the SFRs in that ST using a functional and interface specification and guidance documentation, to understand the security behaviour.

151 The analysis is supported by independent testing of the TSF.

152 This EAL provides a meaningful increase in assurance over unevaluated IT.

Assurance components
ASE_CCL.1 Conformance claims
ASE_ECD.1 Extended components definition
ASE_INT.1 ST introduction
ASE_REQ.1 Stated security requirements
ASE_TSS.1 TOE summary specification
ACM_CAP.1 Version numbers
ADO_IGS.1 Installation, generation, and start-up procedures
ADV_FSP.1 Informal functional specification
ADV_RCR.1 Informal correspondence demonstration
AGD_ADM.1 Administrator guidance
AGD_USR.1 User guidance
ATE_IND.1 Independent testing - conformance

Table 3 EAL1

4.4 Evaluation assurance level 2 (EAL2) - structurally tested

Objectives

153 EAL2 requires the co-operation of the developer in terms of the delivery of design information and test results, but should not demand more effort on the part of the developer than is consistent with good commercial practice. As such it should not require a substantially increased investment of cost or time.

154 EAL2 is therefore applicable in those circumstances where developers or users require a low to moderate level of independently assured security in the absence of ready availability of the complete development record. Such a situation may arise when securing legacy systems, or where access to the developer may be limited.

Assurance components

155 EAL2 provides assurance by a full security target and an analysis of the SFRs in that ST, using a functional and interface specification, guidance documentation and the high-level design of the TOE, to understand the security behaviour.

156 The analysis is supported by independent testing of the TSF, evidence of developer testing based on the functional specification, selective independent confirmation of the developer test results, and evidence of a developer search for obvious vulnerabilities (e.g. those in the public domain).

157 EAL2 also provides assurance through a configuration list for the TOE, and evidence of secure delivery procedures.

158 This EAL represents a meaningful increase in assurance from EAL1 by requiring developer testing, a vulnerability analysis, and independent testing based upon more detailed TOE specifications.

Assurance components
ASE_CCL.1 Conformance claims
ASE_ECD.1 Extended components definition
ASE_INT.1 ST introduction
ASE_OBJ.1 Security objectives
ASE_REQ.2 Derived security requirements
ASE_SPD.1 Security problem definition
ASE_TSS.1 TOE summary specification
ACM_CAP.2 Configuration items
ADO_DEL.1 Delivery procedures
ADO_IGS.1 Installation, generation, and start-up procedures
ADV_FSP.1 Informal functional specification
ADV_HLD.1 Descriptive high-level design
ADV_RCR.1 Informal correspondence demonstration

Assurance components
AGD_ADM.1 Administrator guidance
AGD_USR.1 User guidance
ATE_COV.1 Evidence of coverage
ATE_FUN.1 Functional testing
ATE_IND.2 Independent testing - sample
AVA_VLA.1 Developer vulnerability analysis

Table 4 EAL2

4.5 Evaluation assurance level 3 (EAL3) - methodically tested and checked

Objectives

159 EAL3 permits a conscientious developer to gain maximum assurance from positive security engineering at the design stage without substantial alteration of existing sound development practices.

160 EAL3 is applicable in those circumstances where developers or users require a moderate level of independently assured security, and require a thorough investigation of the TOE and its development without substantial re-engineering.

Assurance components

161 EAL3 provides assurance by a full security target and an analysis of the SFRs in that ST, using a functional and interface specification, guidance documentation, and the high-level design of the TOE, to understand the security behaviour.

162 The analysis is supported by independent testing of the TSF, evidence of developer testing based on the functional specification and high-level design, selective independent confirmation of the developer test results, and evidence of a developer search for obvious vulnerabilities (e.g. those in the public domain).The analysis is supported by independent testing of the TSF, evidence of developer testing based on the functional specification and high-level design, selective independent confirmation of the developer test results, and evidence of a developer search for obvious vulnerabilities (e.g. those in the public domain).

163 EAL3 also provides assurance through the use of development environment controls, TOE configuration management, and evidence of secure delivery procedures.

164 This EAL represents a meaningful increase in assurance from EAL2 by requiring more complete testing coverage of the security functions and mechanisms and/or procedures that provide some confidence that the TOE will not be tampered with during development.

Assurance components
ASE_CCL.1 Conformance claims
ASE_ECD.1 Extended components definition
ASE_INT.1 ST introduction
ASE_OBJ.1 Security objectives
ASE_REQ.2 Derived security requirements
ASE_SPD.1 Security problem definition
ASE_TSS.1 TOE summary specification
ACM_CAP.3 Authorisation controls
ACM_SCP.1 TOE CM coverage
ADO_DEL.1 Delivery procedures
ADO_IGS.1 Installation, generation, and start-up procedures
ADV_FSP.1 Informal functional specification
ADV_HLD.2 Security enforcing high-level design
ADV_RCR.1 Informal correspondence demonstration
AGD_ADM.1 Administrator guidance
AGD_USR.1 User guidance
ALC_DVS.1 Identification of security measures
ATE_COV.2 Analysis of coverage
ATE_DPT.1 Testing: high-level design
ATE_FUN.1 Functional testing
ATE_IND.2 Independent testing - sample
AVA_MSU.1 Examination of guidance
AVA_VLA.1 Developer vulnerability analysis

Table 5 EAL3

4.6 Evaluation assurance level 4 (EAL4) - methodically designed, tested, and reviewed

Objectives

165 EAL4 permits a developer to gain maximum assurance from positive security engineering based on good commercial development practices which, though rigorous, do not require substantial specialist knowledge, skills, and other resources. EAL4 is the highest level at which it is likely to be economically feasible to retrofit to an existing product line.

166 EAL4 is therefore applicable in those circumstances where developers or users require a moderate to high level of independently assured security in conventional commodity TOEs and are prepared to incur additional security-specific engineering costs.

Assurance components

167 EAL4 provides assurance by a full security target and an analysis of the SFRs in that ST, using a functional and complete interface specification, guidance documentation, the high-level and low-level design of the TOE, and a subset of the implementation, to understand the security behaviour.

Assurance is additionally gained through an informal model of the TOE security policy.

168 The analysis is supported by independent testing of the TSF, evidence of developer testing based on the functional specification and high-level design, selective independent confirmation of the developer test results, evidence of a developer search for vulnerabilities, and an independent vulnerability analysis demonstrating resistance to penetration attackers with a low attack potential

169 EAL4 also provides assurance through the use of development environment controls and additional TOE configuration management including automation, and evidence of secure delivery procedures.

170 This EAL represents a meaningful increase in assurance from EAL3 by requiring more design description, a subset of the implementation, and improved mechanisms and/or procedures that provide confidence that the TOE will not be tampered with during development or delivery.

Assurance components
ASE_CCL.1 Conformance claims
ASE_ECD.1 Extended components definition
ASE_INT.1 ST introduction
ASE_OBJ.1 Security objectives
ASE_REQ.2 Derived security requirements
ASE_SPD.1 Security problem definition
ASE_TSS.1 TOE summary specification
ACM_AUT.1 Partial CM automation
ACM_CAP.4 Generation support and acceptance procedures
ACM_SCP.2 Problem tracking CM coverage
ADO_DEL.2 Detection of modification
ADO_IGS.1 Installation, generation, and start-up procedures
ADV_FSP.2 Fully defined external interfaces
ADV_HLD.2 Security enforcing high-level design
ADV_IMP.1 Subset of the implementation of the TSF
ADV_LLD.1 Descriptive low-level design
ADV_RCR.1 Informal correspondence demonstration
ADV_SPM.1 Informal TOE security policy model
AGD_ADM.1 Administrator guidance
AGD_USR.1 User guidance
ALC_DVS.1 Identification of security measures
ALC_LCD.1 Developer defined life-cycle model
ALC_TAT.1 Well-defined development tools
ATE_COV.2 Analysis of coverage
ATE_DPT.1 Testing: high-level design
ATE_FUN.1 Functional testing
ATE_IND.2 Independent testing - sample
AVA_MSU.2 Validation of analysis
AVA_VLA.2 Independent vulnerability analysis

Table 6 EAL4

4.7 Evaluation assurance level 5 (EAL5) - semiformally designed and tested

Objectives

171 EAL5 permits a developer to gain maximum assurance from security engineering based upon rigorous commercial development practices supported by moderate application of specialist security engineering techniques. Such a TOE will probably be designed and developed with the intent of achieving EAL5 assurance. It is likely that the additional costs attributable to the EAL5 requirements, relative to rigorous development without the application of specialised techniques, will not be large.

172 EAL5 is therefore applicable in those circumstances where developers or users require a high level of independently assured security in a planned development and require a rigorous development approach without incurring unreasonable costs attributable to specialist security engineering techniques.

Assurance components

173 EAL5 provides assurance by a full security target and an analysis of the SFRs in that ST, using a functional and complete interface specification, guidance documentation, the high-level and low-level design of the TOE, and all of the implementation, to understand the security behaviour. Assurance is additionally gained through a formal model of the TOE security policy and a semiformal presentation of the functional specification and high-level design and a semiformal demonstration of correspondence between them. A modular TSF design is also required.

174 The analysis is supported by independent testing of the TSF, evidence of developer testing based on the functional specification, high-level design and low-level design, selective independent confirmation of the developer test results, evidence of a developer search for vulnerabilities, and an independent vulnerability analysis demonstrating resistance to penetration attackers with a moderate attack potential. The analysis also includes validation of the developer?s covert channel analysis.

175 EAL5 also provides assurance through the use of a development environment controls, and comprehensive TOE configuration management including automation, and evidence of secure delivery procedures.

176 This EAL represents a meaningful increase in assurance from EAL4 by requiring semiformal design descriptions, the entire implementation, a more structured (and hence analysable) architecture, covert channel analysis, and improved mechanisms and/or procedures that provide confidence that the TOE will not be tampered with during development.

Assurance components
ASE_CCL.1 Conformance claims
ASE_ECD.1 Extended components definition
ASE_INT.1 ST introduction

Assurance components
ASE_OBJ.1 Security objectives
ASE_REQ.2 Derived security requirements
ASE_SPD.1 Security problem definition
ASE_TSS.1 TOE summary specification
ACM_AUT.1 Partial CM automation
ACM_CAP.4 Generation support and acceptance procedures
ACM_SCP.3 Development tools CM coverage
ADO_DEL.2 Detection of modification
ADO_IGS.1 Installation, generation, and start-up procedures
ADV_FSP.3 Semiformal functional specification
ADV_HLD.3 Semiformal high-level design
ADV_IMP.2 Implementation of the TSF
ADV_INT.1 Modularity
ADV_LLD.1 Descriptive low-level design
ADV_RCR.2 Semiformal correspondence demonstration
ADV_SPM.3 Formal TOE security policy model
AGD_ADM.1 Administrator guidance
AGD_USR.1 User guidance
ALC_DVS.1 Identification of security measures
ALC_LCD.2 Standardised life-cycle model
ALC_TAT.2 Compliance with implementation standards
ATE_COV.2 Analysis of coverage
ATE_DPT.2 Testing: low-level design
ATE_FUN.1 Functional testing
ATE_IND.2 Independent testing - sample
AVA_CCA.1 Covert channel analysis
AVA_MSU.2 Validation of analysis
AVA_VLA.3 Moderately resistant

Table 7 EAL5

4.8 Evaluation assurance level 6 (EAL6) - semiformally verified design and tested

Objectives

177 EAL6 permits developers to gain high assurance from application of security engineering techniques to a rigorous development environment in order to produce a premium TOE for protecting high value assets against significant risks.

178 EAL6 is therefore applicable to the development of security TOEs for application in high risk situations where the value of the protected assets justifies the additional costs.

Assurance components

179 EAL6 provides assurance by a full security target and an analysis of the SFRs in that ST, using a functional and complete interface specification,

guidance documentation, the high-level and low-level design of the TOE, and a structured presentation of the implementation, to understand the security behaviour. Assurance is additionally gained through a formal model of the TOE security policy, a semiformal presentation of the functional specification, high-level design, and low-level design and a semiformal demonstration of correspondence between them. A modular and layered TSF design is also required.

180 The analysis is supported by independent testing of the TSF, evidence of developer testing based on the functional specification, high-level design and low-level design, selective independent confirmation of the developer test results, evidence of a developer search for vulnerabilities, and an independent vulnerability analysis demonstrating resistance to penetration attackers with a high attack potential. The analysis also includes validation of the developer?s systematic covert channel analysis.

181 EAL6 also provides assurance through the use of a structured development process, development environment controls, and comprehensive TOE configuration management including complete automation, and evidence of secure delivery procedures.

182 This EAL represents a meaningful increase in assurance from EAL5 by requiring more comprehensive analysis, a structured representation of the implementation, more architectural structure (e.g. layering), more comprehensive independent vulnerability analysis, systematic covert channel identification, and improved configuration management and development environment controls.

Assurance components
ASE_CCL.1 Conformance claims
ASE_ECD.1 Extended components definition
ASE_INT.1 ST introduction
ASE_OBJ.1 Security objectives
ASE_REQ.2 Derived security requirements
ASE_SPD.1 Security problem definition
ASE_TSS.1 TOE summary specification
ACM_AUT.2 Complete CM automation
ACM_CAP.5 Advanced support
ACM_SCP.3 Development tools CM coverage
ADO_DEL.2 Detection of modification
ADO_IGS.1 Installation, generation, and start-up procedures
ADV_FSP.3 Semiformal functional specification
ADV_HLD.4 Semiformal high-level explanation
ADV_IMP.3 Structured implementation of the TSF
ADV_INT.2 Reduction of complexity
ADV_LLD.2 Semiformal low-level design
ADV_RCR.2 Semiformal correspondence demonstration
ADV_SPM.3 Formal TOE security policy model
AGD_ADM.1 Administrator guidance
AGD_USR.1 User guidance

Assurance components
ALC_DVS.2 Sufficiency of security measures
ALC_LCD.2 Standardised life-cycle model
ALC_TAT.3 Compliance with implementation standards - all parts
ATE_COV.3 Rigorous analysis of coverage
ATE_DPT.2 Testing: low-level design
ATE_FUN.2 Ordered functional testing
ATE_IND.2 Independent testing - sample
AVA_CCA.2 Systematic covert channel analysis
AVA_MSU.3 Analysis and testing for insecure states
AVA_VLA.4 Highly resistant

Table 8 EAL6

4.9 Evaluation assurance level 7 (EAL7) - formally verified design and tested

Objectives

183 EAL7 is applicable to the development of security TOEs for application in extremely high risk situations and/or where the high value of the assets justifies the higher costs. Practical application of EAL7 is currently limited to TOEs with tightly focused security functionality that is amenable to extensive formal analysis.

Assurance components

184 EAL7 provides assurance by a full security target and an analysis of the SFRs in that ST, using a functional and complete interface specification, guidance documentation, the high-level and low-level design of the TOE, and a structured presentation of the implementation, to understand the security behaviour. Assurance is additionally gained through a formal model of the TOE security policy, a formal presentation of the functional specification and high-level design, a semiformal presentation of the low-level design, and formal and semiformal demonstration of correspondence between them, as appropriate. A modular, layered and simple TSF design is also required.

185 The analysis is supported by independent testing of the TSF, evidence of developer testing based on the functional specification high-level design, low-level design and implementation representation, complete independent confirmation of the developer test results, evidence of a developer search for vulnerabilities, and an independent vulnerability analysis demonstrating resistance to penetration attackers with a high attack potential. The analysis also includes validation of the developer?s systematic covert channel analysis.

186 EAL7 also provides assurance through the use of a structured development process, development environment controls, and comprehensive TOE

configuration management including complete automation, and evidence of secure delivery procedures.

187 This EAL represents a meaningful increase in assurance from EAL6 by requiring more comprehensive analysis using formal representations and formal correspondence, and comprehensive testing.

Assurance components
ASE_CCL.1 Conformance claims
ASE_ECD.1 Extended components definition
ASE_INT.1 ST introduction
ASE_OBJ.1 Security objectives
ASE_REQ.2 Derived security requirements
ASE_SPD.1 Security problem definition
ASE_TSS.1 TOE summary specification
ACM_AUT.2 Complete CM automation
ACM_CAP.5 Advanced support
ACM_SCP.3 Development tools CM coverage
ADO_DEL.3 Prevention of modification
ADO_IGS.1 Installation, generation, and start-up procedures
ADV_FSP.4 Formal functional specification
ADV_HLD.5 Formal high-level design
ADV_IMP.3 Structured implementation of the TSF
ADV_INT.3 Minimisation of complexity
ADV_LLD.2 Semiformal low-level design
ADV_RCR.3 Formal correspondence demonstration
ADV_SPM.3 Formal TOE security policy model
AGD_ADM.1 Administrator guidance
AGD_USR.1 User guidance
ALC_DVS.2 Sufficiency of security measures
ALC_LCD.3 Measurable life-cycle model
ALC_TAT.3 Compliance with implementation standards - all parts
ATE_COV.3 Rigorous analysis of coverage
ATE_DPT.3 Testing: implementation representation
ATE_FUN.2 Ordered functional testing
ATE_IND.3 Independent testing - complete
AVA_CCA.2 Systematic covert channel analysis
AVA_MSU.3 Analysis and testing for insecure states
AVA_VLA.4 Highly resistant

Table 9 EAL7

5 Assurance classes, families, and components

188 The next seven clauses provide the detailed requirements, presented in alphabetical order, of each of the assurance components, grouped by class and family.

6 Class ACM: Configuration management

189 Configuration management (CM) is one means for establishing that the TOE meets the SFRs. CM establishes this by requiring discipline and control in the processes of refinement and modification of the TOE and the related information. CM systems are put in place to ensure the integrity of the portions of the TOE that they control, by providing a method of tracking any changes, and by ensuring that all changes are authorised.

190 Figure **7** shows the families within this class, and the hierarchy of components within the families.

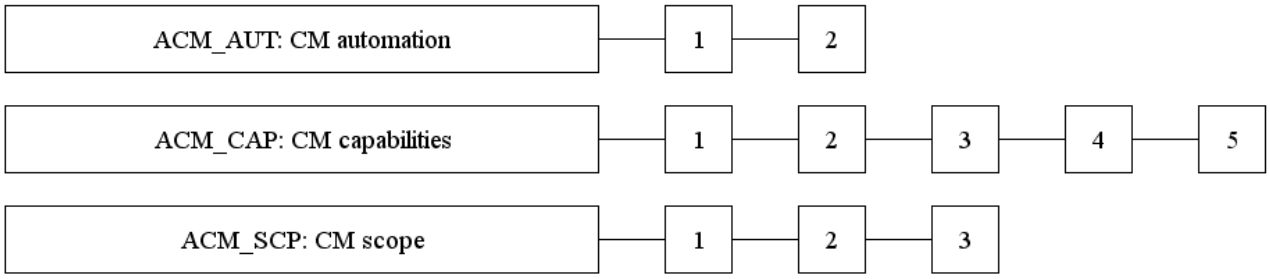

Figure 7 - ACM: Configuration management class decomposition

6.1 CM automation (ACM_AUT)

Objectives

191 The objective of introducing automated CM tools is to increase the effectiveness of the CM system. While both automated and manual CM systems can be bypassed, ignored, or prove insufficient to prevent unauthorised modification, automated systems are less susceptible to human error or negligence.

Component levelling

192 The components in this family are levelled on the basis of the set of configuration items that are controlled through automated means.

Application notes

193 ACM_AUT.1.1C introduces a requirement that is related to the implementation representation of the TOE. The implementation representation of the TOE consists of all hardware, software, and firmware that comprise the physical TOE. In the case of a software-only TOE, the implementation representation may consist solely of source and object code.

194 ACM_AUT.1.2C introduces a requirement that the CM system provide an automated means to support the generation of the TOE. This requires that the CM system provide an automated means to assist in determining that the correct configuration items are used in generating the TOE.

195 ACM_AUT.2.5C introduces a requirement that the CM system provide an automated means to ascertain the changes between the TOE and its preceding version. If no previous version of the TOE exists, the developer still needs to provide an automated means to ascertain the changes between the TOE and a future version of the TOE.

ACM_AUT.1 Partial CM automation

Dependencies

ACM_CAP.3 Authorisation controls

Objectives

196 In development environments where the implementation representation is complex or is being developed by multiple developers, it is difficult to control changes without the support of automated tools. In particular, these automated tools need to be able to support the numerous changes that occur during development and ensure that those changes are authorised. It is the objective of this component to ensure that the implementation representation is controlled through automated means.

Developer action elements

ACM_AUT.1.1D The developer shall use a CM system.

ACM_AUT.1.2D The developer shall provide a CM plan.

Content and presentation of evidence elements

ACM_AUT.1.1C The CM system shall provide an automated means by which only authorised changes are made to the TOE implementation representation.

ACM_AUT.1.2C The CM system shall provide an automated means to support the generation of the TOE.

ACM_AUT.1.3C The CM plan shall describe the automated tools used in the CM system.

ACM_AUT.1.4C The CM plan shall describe how the automated tools are used in the CM system.

Evaluator action elements

ACM_AUT.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ACM_AUT.2 Complete CM automation

Dependencies

ACM_CAP.3 Authorisation controls

Objectives

197 In development environments where the configuration items are complex or are being developed by multiple developers, it is difficult to control changes without the support of automated tools. In particular, these automated tools need to be able to support the numerous changes that occur during development and ensure that those changes are authorised. It is the objective of this component to ensure that all configuration items are controlled through automated means.

198 Providing an automated means of ascertaining changes between versions of the TOE and identifying which configuration items are affected by modifications to other configuration items assists in determining the impact of the changes between successive versions of the TOE. This in turn can provide valuable information in determining whether changes to the TOE result in all configuration items being consistent with one another.

Developer action elements

ACM_AUT.2.1D The developer shall use a CM system.

ACM_AUT.2.2D The developer shall provide a CM plan.

Content and presentation of evidence elements

ACM_AUT.2.1C The CM system shall provide an automated means by which only authorised changes are made to the TOE implementation representation, and to all other configuration items.

ACM_AUT.2.2C The CM system shall provide an automated means to support the generation of the TOE.

ACM_AUT.2.3C The CM plan shall describe the automated tools used in the CM system.

ACM_AUT.2.4C The CM plan shall describe how the automated tools are used in the CM system.

ACM_AUT.2.5C The CM system shall provide an automated means to ascertain the changes between the TOE and its preceding version.

ACM_AUT.2.6C The CM system shall provide an automated means to identify all other configuration items that are affected by the modification of a given configuration item.

Evaluator action elements

ACM_AUT.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

6.2 CM capabilities (ACM_CAP)

Objectives

199 The capabilities of the CM system address the likelihood that accidental or unauthorised modifications of the configuration items will occur. The CM system should ensure the integrity of the TOE from the early design stages through all subsequent maintenance efforts.

200 The objectives of this family include the following:

a) ensuring that the TOE is correct and complete before it is sent to the consumer;

b) ensuring that no configuration items are missed during evaluation;

c) preventing unauthorised modification, addition, or deletion of TOE configuration items.

201 In the case where the TOE is a subset of a product, the ACM requirements apply only to the TOE configuration items, not to the product as a whole. While it is desired that CM be applied from the early design stages and continue into the future, ACM requires that CM be in place and in use prior to the end of the evaluation.

Component levelling

202 The components in this family are levelled on the basis of the CM system capabilities, the scope of the CM documentation provided by the developer, and whether the developer provides justification that the CM system meets its security requirements.

Application notes

203 ACM_CAP.2 Configuration items introduces several elements which refer to configuration items. The CM scope (ACM_SCP) family contains requirements for the configuration items to be tracked by the CM system.

204 ACM_CAP.2.3C introduces a requirement that a configuration list be provided. The configuration list contains all configuration items that are maintained by the CM system.

205 ACM_CAP.2.7C introduces a requirement that the CM system uniquely identify all configuration items. This also requires that modifications to configuration items result in a new, unique identifier being assigned.

206 ACM_CAP.3.9C introduces the requirement that the evidence shall demonstrate that the CM system operates in accordance with the CM plan. Examples of such evidence might be documentation such as screen snapshots or audit trail output from the CM system, or a detailed demonstration of the CM system by the developer. The evaluator is responsible for determining that this

evidence is sufficient to show that the CM system operates in accordance with the CM plan.

207 ACM_CAP.3.10C introduces the requirement that evidence be provided to show that all configuration items are being maintained under the CM system. Since a configuration item refers to an item that is on the configuration list, this requirement states that all items on the configuration list are maintained under the CM system.

208 ACM_CAP.4.12C introduces the requirement that the CM system support the generation of the TOE. This requires that the CM system provide information and/or electronic means to assist in determining that the correct configuration items are used in generating the TOE.

209 CM capabilities (ACM_CAP) identifies the CM requirements to be imposed on all items identified in the configuration item list. Other than the TOE itself, CM capabilities (ACM_CAP) leaves the contents of the configuration item list to the discretion of the developer. (CM scope (ACM_SCP) can be used to identify specific items that must be included in the configuration item list, and hence covered by CM.)

ACM_CAP.1 Version numbers

Objectives

210 A unique reference is required to ensure that there is no ambiguity in terms of which instance of the TOE is being evaluated. Labelling the TOE with its reference ensures that users of the TOE can be aware of which instance of the TOE they are using.

Developer action elements

ACM_CAP.1.1D The developer shall provide a reference for the TOE.

Content and presentation of evidence elements

ACM_CAP.1.1C The reference for the TOE shall be unique to each version of the TOE.

ACM_CAP.1.2C The TOE shall be labelled with its reference.

Evaluator action elements

ACM_CAP.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ACM_CAP.2 Configuration items

Objectives

211 A unique reference is required to ensure that there is no ambiguity in terms of which instance of the TOE is being evaluated. Labelling the TOE with its

reference ensures that users of the TOE can be aware of which instance of the TOE they are using.

212 Unique identification of the configuration items leads to a clearer understanding of the composition of the TOE, which in turn helps to determine those items which are subject to the evaluation requirements for the TOE.

Developer action elements

ACM_CAP.2.1D The developer shall provide a reference for the TOE.

ACM_CAP.2.2D The developer shall use a CM system.

ACM_CAP.2.3D The developer shall provide CM documentation.

Content and presentation of evidence elements

ACM_CAP.2.1C The reference for the TOE shall be unique to each version of the TOE.

ACM_CAP.2.2C The TOE shall be labelled with its reference.

ACM_CAP.2.3C The CM documentation shall include a configuration list.

ACM_CAP.2.4C The configuration list shall uniquely identify all configuration items that comprise the TOE.

ACM_CAP.2.5C The configuration list shall describe the configuration items that comprise the TOE.

ACM_CAP.2.6C The CM documentation shall describe the method used to uniquely identify the configuration items.

ACM_CAP.2.7C The CM system shall uniquely identify all configuration items.

Evaluator action elements

ACM_CAP.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ACM_CAP.3 Authorisation controls

Dependencies

ALC_DVS.1 Identification of security measures

Objectives

213 A unique reference is required to ensure that there is no ambiguity in terms of which instance of the TOE is being evaluated. Labelling the TOE with its reference ensures that users of the TOE can be aware of which instance of the TOE they are using.

214 Unique identification of the configuration items leads to a clearer understanding of the composition of the TOE, which in turn helps to determine those items which are subject to the evaluation requirements for the TOE.

215 Providing controls to ensure that unauthorised modifications are not made to the TOE, and ensuring proper functionality and use of the CM system, helps to maintain the integrity of the TOE.

Developer action elements

ACM_CAP.3.1D The developer shall provide a reference for the TOE.

ACM_CAP.3.2D The developer shall use a CM system.

ACM_CAP.3.3D The developer shall provide CM documentation.

Content and presentation of evidence elements

ACM_CAP.3.1C The reference for the TOE shall be unique to each version of the TOE.

ACM_CAP.3.2C The TOE shall be labelled with its reference.

ACM_CAP.3.3C The CM documentation shall include a configuration list and a CM plan.

ACM_CAP.3.4C The configuration list shall uniquely identify all configuration items that comprise the TOE.

ACM_CAP.3.5C The configuration list shall describe the configuration items that comprise the TOE.

ACM_CAP.3.6C The CM documentation shall describe the method used to uniquely identify the configuration items.

ACM_CAP.3.7C The CM system shall uniquely identify all configuration items.

ACM_CAP.3.8C The CM plan shall describe how the CM system is used.

ACM_CAP.3.9C The evidence shall demonstrate that the CM system is operating in accordance with the CM plan.

ACM_CAP.3.10C The CM documentation shall provide evidence that all configuration items have been and are being effectively maintained under the CM system.

ACM_CAP.3.11C The CM system shall provide measures such that only authorised changes are made to the configuration items.

Evaluator action elements

ACM_CAP.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ACM_CAP.4 Generation support and acceptance procedures

Dependencies

ALC_DVS.1 Identification of security measures

Objectives

216 A unique reference is required to ensure that there is no ambiguity in terms of which instance of the TOE is being evaluated. Labelling the TOE with its reference ensures that users of the TOE can be aware of which instance of the TOE they are using.

217 Unique identification of the configuration items leads to a clearer understanding of the composition of the TOE, which in turn helps to determine those items which are subject to the evaluation requirements for the TOE.

218 Providing controls to ensure that unauthorised modifications are not made to the TOE, and ensuring proper functionality and use of the CM system, helps to maintain the integrity of the TOE.

219 The purpose of acceptance procedures is to confirm that any creation or modification of configuration items is authorised.

Developer action elements

ACM_CAP.4.1D The developer shall provide a reference for the TOE.

ACM_CAP.4.2D The developer shall use a CM system.

ACM_CAP.4.3D The developer shall provide CM documentation.

Content and presentation of evidence elements

ACM_CAP.4.1C The reference for the TOE shall be unique to each version of the TOE.

ACM_CAP.4.2C The TOE shall be labelled with its reference.

ACM_CAP.4.3C The CM documentation shall include a configuration list, a CM plan, and an acceptance plan.

ACM_CAP.4.4C The configuration list shall uniquely identify all configuration items that comprise the TOE.

ACM_CAP.4.5C The configuration list shall describe the configuration items that comprise the TOE.

ACM_CAP.4.6C The CM documentation shall describe the method used to uniquely identify the configuration items.

ACM_CAP.4.7C The CM system shall uniquely identify all configuration items.

ACM_CAP.4.8C The CM plan shall describe how the CM system is used.

ACM_CAP.4.9C The evidence shall demonstrate that the CM system is operating in accordance with the CM plan.

ACM_CAP.4.10C The CM documentation shall provide evidence that all configuration items have been and are being effectively maintained under the CM system.

ACM_CAP.4.11C The CM system shall provide measures such that only authorised changes are made to the configuration items.

ACM_CAP.4.12C The CM system shall support the generation of the TOE.

ACM_CAP.4.13C The acceptance plan shall describe the procedures used to accept modified or newly created configuration items as part of the TOE.

Evaluator action elements

ACM_CAP.4.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ACM_CAP.5 Advanced support

Dependencies

ALC_DVS.2 Sufficiency of security measures

Objectives

220 A unique reference is required to ensure that there is no ambiguity in terms of which instance of the TOE is being evaluated. Labelling the TOE with its reference ensures that users of the TOE can be aware of which instance of the TOE they are using.

221 Unique identification of the configuration items leads to a clearer understanding of the composition of the TOE, which in turn helps to determine those items which are subject to the evaluation requirements for the TOE.

222 Providing controls to ensure that unauthorised modifications are not made to the TOE, and ensuring proper functionality and use of the CM system, helps to maintain the integrity of the TOE.

223 The purpose of acceptance procedures is to confirm that any creation or modification of configuration items is authorised.

224 Integration procedures help to ensure that generation of the TOE from a managed set of configuration items is correctly performed in an authorised manner.

225 Requiring that the CM system be able to identify the master copy of the material used to generate the TOE helps to ensure that the integrity of this material is preserved by the appropriate technical, physical and procedural safeguards.

Developer action elements

ACM_CAP.5.1D The developer shall provide a reference for the TOE.

ACM_CAP.5.2D The developer shall use a CM system.

ACM_CAP.5.3D The developer shall provide CM documentation.

Content and presentation of evidence elements

ACM_CAP.5.1C The reference for the TOE shall be unique to each version of the TOE.

ACM_CAP.5.2C The TOE shall be labelled with its reference.

ACM_CAP.5.3C The CM documentation shall include a configuration list, a CM plan, an acceptance plan, and integration procedures.

ACM_CAP.5.4C The configuration list shall describe the configuration items that comprise the TOE.

ACM_CAP.5.5C The CM documentation shall describe the method used to uniquely identify the configuration items.

ACM_CAP.5.6C The CM system shall uniquely identify all configuration items.

ACM_CAP.5.7C The CM plan shall describe how the CM system is used.

ACM_CAP.5.8C The evidence shall demonstrate that the CM system is operating in accordance with the CM plan.

ACM_CAP.5.9C The CM documentation shall provide evidence that all configuration items have been and are being effectively maintained under the CM system.

ACM_CAP.5.10C The CM system shall provide measures such that only authorised changes are made to the configuration items.

ACM_CAP.5.11C The CM system shall support the generation of the TOE.

ACM_CAP.5.12C The acceptance plan shall describe the procedures used to accept modified or newly created configuration items as part of the TOE.

ACM_CAP.5.13C The integration procedures shall describe how the CM system is applied in the TOE manufacturing process.

ACM_CAP.5.14C The CM system shall require that the person responsible for accepting a configuration item into CM is not the person who developed it.

ACM_CAP.5.15C The CM system shall clearly identify the configuration items that comprise the TSF.

ACM_CAP.5.16C The CM system shall support the audit of all modifications to the TOE, including the originator, date, and time in the audit trail.

ACM_CAP.5.17C The CM system shall be able to identify the master copy of all material used to generate the TOE.

ACM_CAP.5.18C The CM documentation shall demonstrate that the use of the CM system, together with the development security measures, allow only authorised changes to be made to the TOE.

ACM_CAP.5.19C The CM documentation shall demonstrate that the use of the integration procedures ensures that the generation of the TOE is correctly performed in an authorised manner.

ACM_CAP.5.20C The CM documentation shall demonstrate that the CM system is sufficient to ensure that the person responsible for accepting a configuration item into CM is not the person who developed it.

ACM_CAP.5.21C The CM documentation shall justify that the acceptance procedures provide for an adequate and appropriate review of changes to all configuration items.

Evaluator action elements

ACM_CAP.5.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

6.3 CM scope (ACM_SCP)

Objectives

226 The objective of this family is to require items to be included as configuration items and hence placed under the CM requirements of CM capabilities (ACM_CAP). Applying configuration management to these additional items provides additional assurance that the integrity of TOE is maintained.

Component levelling

227 The components in this family are levelled on the basis of which of the following are required to be included as configuration items: implementation representation; the evaluation evidence required by the assurance components in the ST; security flaws; and development tools and related information.

Application notes

228 While CM capabilities (ACM_CAP) mandates a list of configuration items and that each item on this list be under CM, other than the TOE itself, CM capabilities (ACM_CAP) leaves the contents of the configuration item list to the discretion of the developer. CM scope (ACM_SCP) narrows this discretion by identifying items that must be included in the configuration item list, and hence come under the CM requirements of CM capabilities (ACM_CAP).

229 ACM_SCP.1.1C introduces the requirement that the TOE implementation representation be included in the list of configuration items. The TOE implementation representation refers to all hardware, software, and firmware that comprise the physical TOE. In the case of a software-only TOE, the implementation representation may consist solely of source and object code.

230 ACM_SCP.1.1C also introduces the requirement that the evaluation evidence required by the other assurance components in the ST be included in the list of configuration items.

231 ACM_SCP.2.1C introduces the requirement that security flaws be included in the list of configuration items. This requires that information regarding previous security flaws and their resolution be maintained, as well as details regarding current security flaws.

232 ACM_SCP.3.1C introduces the requirement that development tools and other related information be included in the list of configuration items. Examples of development tools are programming languages and compilers. Information pertaining to TOE generation items (such as compiler options, installation/generation options, and build options) is an example of information relating to development tools.

ACM_SCP.1 TOE CM coverage

Dependencies

ACM_CAP.3 Authorisation controls

Objectives

233 A CM system can control changes only to those items that have been placed under CM (i.e., the configuration items identified in the configuration item list). Placing the TOE implementation and the evaluation evidence required by the other SARs in the ST under CM provides assurance that they have been modified in a controlled manner with proper authorisations.

Developer action elements

ACM_SCP.1.1D The developer shall provide a list of configuration items for the TOE.

Content and presentation of evidence elements

ACM_SCP.1.1C The list of configuration items shall include the following: implementation representation and the evaluation evidence required by the SARs in the ST.

Evaluator action elements

ACM_SCP.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ACM_SCP.2 Problem tracking CM coverage

Dependencies

ACM_CAP.3 Authorisation controls

Objectives

234 A CM system can control changes only to those items that have been placed under CM (i.e., the configuration items identified in the configuration item list). Placing the TOE implementation and the evaluation evidence required by the other SARs in the ST under CM provides assurance that they have been modified in a controlled manner with proper authorisations.

235 A CM system can control changes only to those items that have been placed under CM (i.e., the configuration items identified in the configuration item list). Placing the TOE implementation and the evaluation evidence required by the other SARs in the ST under CM provides assurance that they have been modified in a controlled manner with proper authorisations.

Developer action elements

ACM_SCP.2.1D The developer shall provide a list of configuration items for the TOE.

Content and presentation of evidence elements

ACM_SCP.2.1C The list of configuration items shall include the following: implementation representation, security flaws, and the evaluation evidence required by the SARs in the ST.

Evaluator action elements

ACM_SCP.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ACM_SCP.3 Development tools CM coverage

Dependencies

ACM_CAP.3 Authorisation controls

Objectives

236 A CM system can control changes only to those items that have been placed under CM (i.e., the configuration items identified in the configuration item list). Placing the TOE implementation and the evaluation evidence required by the other SARs in the ST under CM provides assurance that they have been modified in a controlled manner with proper authorisations.

237 A CM system can control changes only to those items that have been placed under CM (i.e., the configuration items identified in the configuration item list). Placing the TOE implementation and the evaluation evidence required by the other SARs in the ST under CM provides assurance that they have been modified in a controlled manner with proper authorisations.

238 A CM system can control changes only to those items that have been placed under CM (i.e., the configuration items identified in the configuration item list). Placing the TOE implementation and the evaluation evidence required by the other SARs in the ST under CM provides assurance that they have been modified in a controlled manner with proper authorisations.

Developer action elements

ACM_SCP.3.1D The developer shall provide a list of configuration items for the TOE.

Content and presentation of evidence elements

ACM_SCP.3.1C The list of configuration items shall include the following: implementation representation; security flaws; development tools and related information; and the evaluation evidence required by the SARs in the ST.

Evaluator action elements

ACM_SCP.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

7 Class ADO: Delivery and operation

239 Delivery and operation provides requirements for correct delivery, installation, generation, and start-up of the TOE.

240 Figure **8** shows the families within this class, and the hierarchy of components within the families.

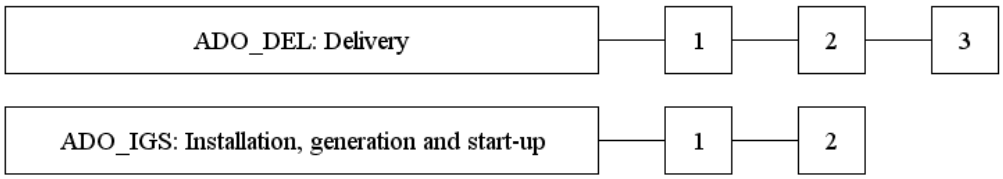

Figure 8 - ADO: Delivery and operation class decomposition

7.1 Delivery (ADO_DEL)

Objectives

241 The requirements for delivery call for system control and distribution facilities and procedures that detail the measures necessary to provide assurance that the security of the TOE is maintained during distribution of the TOE. For a valid distribution of the TOE, the procedures used for the distribution of the TOE address the threats identified in the PP/ST relating to the security of the TOE during delivery.

Component levelling

242 The components in this family are levelled on the basis of increasing requirements on the developer to maintain security of the TOE during delivery.

Application notes

243 These procedures could consider issues such as:

a) ensuring the TOE received by the consumer corresponds precisely to the TOE Master copy;

b) avoiding/detecting any tampering with the actual version of the TOE;

c) preventing submission of a false version of the TOE;

d) avoiding unwanted knowledge of distribution of the TOE to the consumer;

e) avoiding/detecting the TOE being intercepted during delivery; and

f) avoiding the TOE being delayed or stopped during distribution.

244 Although the procedures consider protection of the TOE in all aspects (integrity, confidentiality, availability), the technical measures introduced in ADO_DEL.2 Detection of modification and ADO_DEL.3 Prevention of modification are required to address integrity issues only.

ADO_DEL.1 Delivery procedures

Developer action elements

ADO_DEL.1.1D The developer shall document procedures for delivery of the TOE or parts of it to the user.

ADO_DEL.1.2D The developer shall use the delivery procedures.

Content and presentation of evidence elements

ADO_DEL.1.1C The delivery documentation shall describe all procedures that are necessary to maintain security when distributing versions of the TOE to a user's site.

Evaluator action elements

ADO_DEL.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADO_DEL.2 Detection of modification

Dependencies

ACM_CAP.3 Authorisation controls

Developer action elements

ADO_DEL.2.1D The developer shall document procedures for delivery of the TOE or parts of it to the user.

ADO_DEL.2.2D The developer shall use the delivery procedures.

Content and presentation of evidence elements

ADO_DEL.2.1C The delivery documentation shall describe all procedures that are necessary to maintain security when distributing versions of the TOE to a user's site.

ADO_DEL.2.2C The delivery documentation shall describe how the various procedures and technical measures provide for the detection of modifications, or any discrepancy between the developer's master copy and the version received at the user site.

ADO_DEL.2.3C The delivery documentation shall describe how the various procedures allow detection of attempts to masquerade as the developer, even in cases in which the developer has sent nothing to the user's site.

Evaluator action elements

ADO_DEL.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADO_DEL.3 Prevention of modification

Dependencies

ACM_CAP.3 Authorisation controls

Developer action elements

ADO_DEL.3.1D The developer shall document procedures for delivery of the TOE or parts of it to the user.

ADO_DEL.3.2D The developer shall use the delivery procedures.

Content and presentation of evidence elements

ADO_DEL.3.1C The delivery documentation shall describe all procedures that are necessary to maintain security when distributing versions of the TOE to a user's site.

ADO_DEL.3.2C The delivery documentation shall describe how the various procedures and technical measures provide for the prevention of modifications, or any discrepancy between the developer's master copy and the version received at the user site.

ADO_DEL.3.3C The delivery documentation shall describe how the various procedures allow detection of attempts to masquerade as the developer, even in cases in which the developer has sent nothing to the user's site.

Evaluator action elements

ADO_DEL.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

7.2 Installation, generation and start-up (ADO_IGS)

Objectives

245 Installation, generation, and start-up procedures are useful for ensuring that the TOE has been installed, generated, and started up in a secure manner as intended by the developer. The requirements for installation, generation and start-up call for a secure transition from the TOE's implementation representation being under configuration control to its initial operation in the user environment.

Component levelling

246 The components in this family are levelled on the basis of whether the TOE generation options are logged.

Application notes

247 It is recognised that the application of these requirements will vary depending on aspects such as whether the TOE is an IT product or system, whether it is delivered in an operational state, or whether it has to be brought up at the TOE owner's site, etc. For a given TOE, there will normally be a division of responsibility with respect to installation, generation and start-up between the TOE developer and the owner of the TOE, but there are examples where all activities take place at one site. For example, for a smart card all aspects of installation, generation and start-up may have been performed at the TOE developer's site. On the other hand the TOE might be delivered as an IT system in the form of software, where all aspects of installation, generation and start-up are carried out at the TOE owner's site.

248 It might also be the case that the TOE is already installed by the time the evaluation starts. In this case it may be inappropriate to demand and analyse installation procedures.

249 Furthermore, the generation requirements are applicable only to TOEs that provide the ability to generate portions of an operational TOE from its implementation representation.

250 The installation, generation, and start-up procedures may exist as a separate documents or could be grouped with other administrative guidance. The requirements in this assurance family are presented separately from those in the Administrator guidance (AGD_ADM) family, due to the infrequent, possibly one-time use of the installation, generation and start-up procedures.

ADO_IGS.1 Installation, generation, and start-up procedures

Dependencies

AGD_ADM.1 Administrator guidance

Developer action elements

ADO_IGS.1.1D The developer shall document procedures necessary for the secure installation, generation, and start-up of the TOE.

Content and presentation of evidence elements

ADO_IGS.1.1C The installation, generation and start-up documentation shall describe all the steps necessary for secure installation, generation and start-up of the TOE.

Evaluator action elements

ADO_IGS.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADO_IGS.1.2E The evaluator shall determine that the installation, generation, and start-up procedures result in a secure configuration.

ADO_IGS.2 Generation log

Dependencies

AGD_ADM.1 Administrator guidance

Developer action elements

ADO_IGS.2.1D The developer shall document procedures necessary for the secure installation, generation, and start-up of the TOE.

Content and presentation of evidence elements

ADO_IGS.2.1C The installation, generation and start-up documentation shall describe all the steps necessary for secure installation, generation and start-up of the TOE.

ADO_IGS.2.2C The installation, generation and start-up documentation shall describe procedures capable of creating a log containing the generation options used to generate the TOE in such a way that it is possible to determine exactly how and when the TOE was generated.

Evaluator action elements

ADO_IGS.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADO_IGS.2.2E The evaluator shall determine that the installation, generation, and start-up procedures result in a secure configuration.

8 Class ADV: Development

251 The development class encompasses four families of requirements for representing the TSF at various levels of abstraction from the functional interface to the implementation representation. The development class also includes a family of requirements for a correspondence mapping between the various TSF representations, ultimately requiring a demonstration of correspondence from the least abstract TSF representation through all intervening TSF representations, with the SFRs provided in the ST. In addition, there is a family of requirements for a TSP model, and for correspondence mappings between the SFRs, the TSP model, and the functional specification. Finally, there is a family of requirements on the internal structure of the TSF, which covers aspects such as modularity, layering, and minimisation of complexity of the TSF.

252 The paradigm evident for these families is one of a functional specification of the TSF, decomposing the TSF into subsystems, decomposing the subsystems into modules, showing the implementation of the modules, and demonstration of correspondence between all decompositions that are provided as evidence. The requirements for the various TSF representations are separated into different families, however, to allow the PP/ST author to specify which subset of the TSF representations are required.

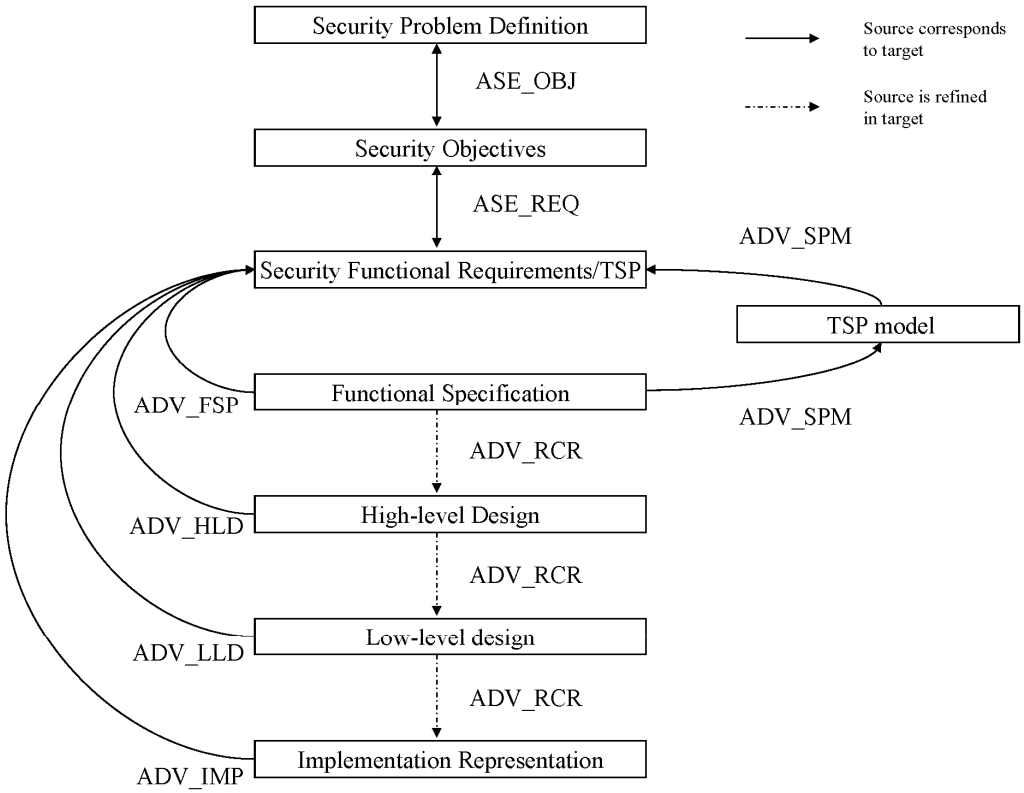

Figure 9 - Relationships between TOE representations and ST entities

253 Figure **9** indicates the relationships between the various TSF representations, the SFRs, the security objectives and the security problem definition. As the figure indicates, the ASE class defines the requirements for the correspondence between the SFRs and the security objectives as well as between the security objectives and the security problem definition.

254 The requirements for all other correspondence shown in Figure **9** are defined in the ADV: Development class. The Security policy modeling (ADV_SPM) family defines the requirements for correspondence between the SFRs and the TSP model, and between the TSP model and the functional specification.

255 The Representation correspondence (ADV_RCR) family defines the requirements for pairwise correspondence between all available TSF representations and the requirements for correspondence between the functional specification and the SFRs.

256 Finally, each assurance family specific to a TSF representation (i.e. Functional specification (ADV_FSP), High-level design (ADV_HLD), Low-level design (ADV_LLD) and Implementation representation (ADV_IMP)) defines requirements relating that TSF representation to the SFRs, the combination of which helps to ensure that the SFRs have been addressed. The traceability analysis is always to be performed from the highest-level

TSF representation down through each of the TSF representations that are provided. The CC captures this traceability requirement via dependencies on the Representation correspondence (ADV_RCR) family.

257 The TSF internals (ADV_INT) family is not represented in this figure, as it is related to the internal structure of the TSF, and is only indirectly related to the process of refinement of the TSF representations.

258 The TOE summary specification (ASE_TSS) family is also not represented in this figure, as it is intended to provide the ST reader with a general overview of how the TOE implements the SFRs, and not as a full TSF representation.

259 The TOE security policy (TSP) is the set of rules that regulate how resources are managed, protected and distributed within a TOE, expressed by the SFRs. The developer is not explicitly required to provide a TSP, as the TSP is expressed by the SFRs, through a combination of security function policies (SFPs) and the other individual requirement elements.

260 The TOE security functions (TSF) are all the parts of the TOE that have to be relied upon for enforcement of the TSP. The TSF includes both parts that directly enforce the TSP, and also those parts that, while not directly enforcing the TSP, contribute to the enforcement of the TSP in a more indirect manner.

261 Although the requirements within several families of this class call for several different TSF representations, it is not absolutely necessary for each and every TSF representation to be in a separate document. Indeed, it may be the case that a single document meets the documentation requirements for more than one TSF representation, since it is the information about each of these TSF representations that is required, rather than the resulting document structure. In cases where multiple TSF representations are combined within a single document, the developer should indicate which documents meet which requirements.

262 Three types of specification style are mandated by this class: informal, semiformal and formal. The functional specification, high-level design, low-level design and TSP models will be written using one or more of these specification styles. Ambiguity in these specifications is reduced by using an increased level of formality.

263 An informal specification is written as prose in natural language. Natural language is used here as meaning communication in any commonly spoken tongue (e.g. Dutch, English, French, German). An informal specification is not subject to any notational or special restrictions other than those required as ordinary conventions for that language (e.g. grammar and syntax). While no notational restrictions apply, the informal specification is also required to provide defined meanings for terms that are used in a context other than that accepted by normal usage.

264 A semiformal specification is written in a restricted syntax language and is typically accompanied by supporting explanatory (informal) prose. The restricted syntax language may be a natural language with restricted sentence structure and keywords with special meanings, or it may be diagrammatic (e.g. data-flow diagrams, state transition diagrams, entity-relationship diagrams, data structure diagrams, and process or program structure diagrams). Whether based on diagrams or natural language, a set of conventions must be supplied to define the restrictions placed on the syntax.

265 A formal specification is written in a notation based upon well-established mathematical concepts, and is typically accompanied by supporting explanatory (informal) prose. These mathematical concepts are used to define the syntax and semantics of the notation and the proof rules that support logical reasoning. The syntactic and semantic rules supporting a formal notation should define how to recognise constructs unambiguously and determine their meaning. There needs to be evidence that it is impossible to derive contradictions, and all rules supporting the notation need to be defined or referenced.

266 Significant assurance can be gained by ensuring that the TSF can be traced through each of its representations, and by ensuring that the TSP model corresponds to the functional specification. The Representation correspondence (ADV_RCR) family contains requirements for correspondence mappings between the various TSF representations, and the Security policy modeling (ADV_SPM) family contains requirements for a correspondence mapping between the TSP model and the functional specification. A correspondence can take the form of an informal demonstration, a semiformal demonstration, or a formal proof.

267 When an informal demonstration of correspondence is required, this means that only a basic correspondence is required. Correspondence methods include, for example, the use of a two-dimensional table with entries denoting correspondence, or the use of appropriate notation of design diagrams. Pointers and references to other documents may also be used.

268 A semiformal demonstration of correspondence requires a structured approach at the analysis of the correspondence. This approach should lessen ambiguity that could exist in an informal correspondence by limiting the interpretation of the terms included in the correspondence. Pointers and references to other documents may be used.

269 A formal proof of correspondence requires that well-established mathematical concepts be used to define the syntax and semantics of the formal notation and the proof rules that support logical reasoning. The security properties need to be expressible in the formal specification language, and these security properties need to be shown to be satisfied by the formal specification. Pointers and references to other documents may also be used.

270 The Representation correspondence (ADV_RCR).*.1C elements require that the developer provide evidence, for each adjacent pair of TSF

representations, that all relevant security functionality of the more abstract TSF representation is refined in the less abstract TSF representation. The Functional specification (ADV_FSP).*.2E, High-level design (ADV_HLD).*.2E, Low-level design (ADV_LLD).*.2E and Implementation representation (ADV_IMP).*.2E elements each require the evaluator to determine that the TSF represented by that family of requirements is an accurate and complete instantiation of the SFRs. In order to determine that a TSF representation is an accurate and complete instantiation of the SFRs, it is intended that the evaluator use the evidence provided by the developer in Representation correspondence (ADV_RCR).*.1C as an input to this determination. By establishing a correspondence between the SFRs and each of successive TSF representations down the chain, this step-wise process will ultimately provide more assurance that the least abstract TSF representation corresponds to the SFRs, which is the ultimate goal of this class. If the evaluator makes no correspondence determinations back to the SFRs for intermediate TSF representations, then trying to determine the correspondence from the least abstract TSF representation back to the SFRs may represent too large a step to be accurately performed. Finally, depending on the set of TSF representations that are required, it is quite possible that the low-level design, high-level design, or even the functional specification might be the least abstract TSF representation that is provided.

271 Figure **10** shows the families within this class, and the hierarchy of components within the families.

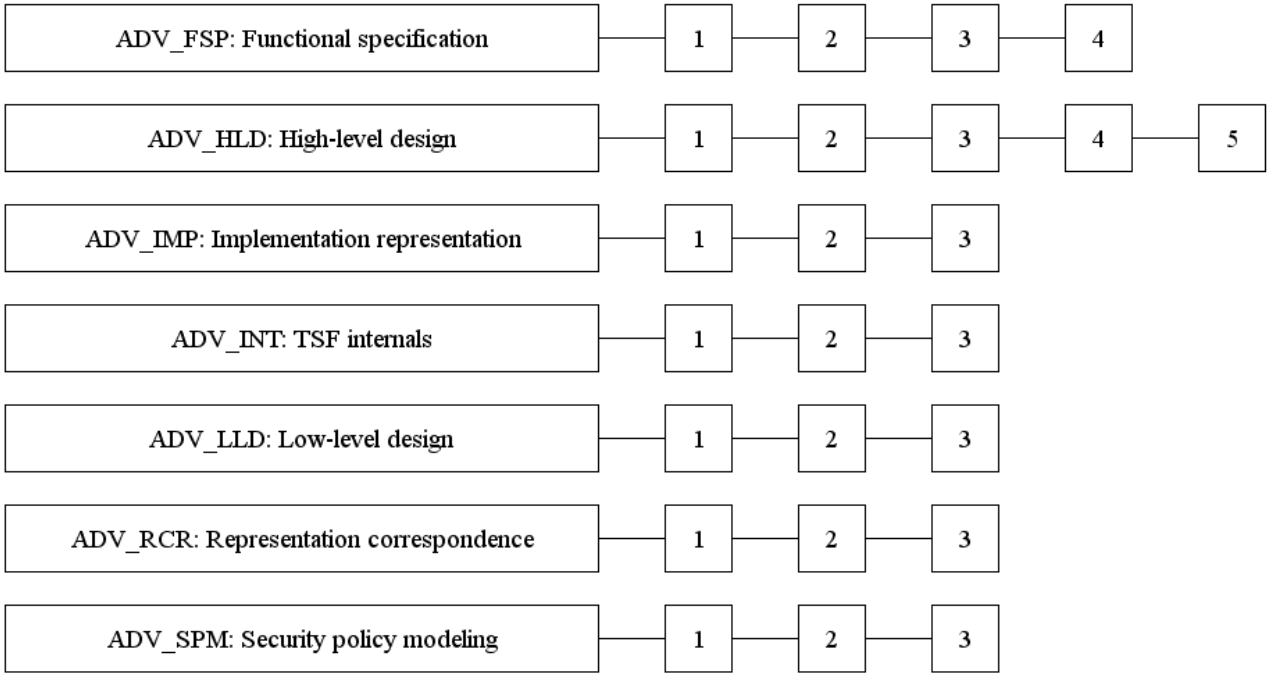

Figure 10 - ADV: Development class decomposition

8.1 Functional specification (ADV_FSP)

Objectives

272 The functional specification is a description of the user-visible interface and behaviour of the TSF. It is an instantiation of the SFRs. The functional specification has to show that all SFRs are addressed.

Component levelling

273 The components in this family are levelled on the basis of the degree of formalism required of the functional specification, and the degree of detail provided for the external interfaces to the TSF.

Application notes

274 The Functional specification (ADV_FSP).*.2E elements within this family define a requirement that the evaluator determine that the functional specification is an accurate and complete instantiation of the SFRs. This provides a direct correspondence between the SFRs and the functional specification, in addition to the pairwise correspondences required by the Representation correspondence (ADV_RCR) family. It is expected that the evaluator will use the evidence provided in Representation correspondence (ADV_RCR) as an input to making this determination, and the requirement for completeness is intended to be relative to the level of abstraction of the functional specification.

275 For ADV_FSP.1.2C, it is intended that sufficient information is provided in the functional specification to understand how the TOE security functional requirements have been addressed, and to enable the specification of tests which reflect the TOE security functional requirements in the ST. It is not necessarily the case that such testing will cover all possible return values and error messages which could be generated at the interface, but the information provided should make clear the results of using an interface in the case of success and the most common instances of failure.

276 ADV_FSP.2.2C introduces a requirement for a complete presentation of the functional interface. This will provide the necessary detail for supporting both thorough testing of the TOE and the assessment of vulnerabilities.

277 In the context of the level of formality of the functional specification, informal, semiformal and formal are considered to be hierarchical in nature. Thus, ADV_FSP.1.1C and ADV_FSP.2.1C may also be met with either a semiformal or formal functional specification, provided that it is supported by informal, explanatory text where appropriate. In addition, ADV_FSP.3.1C may also be met with a formal functional specification.

ADV_FSP.1 Informal functional specification

Dependencies

ADV_RCR.1 Informal correspondence demonstration

Developer action elements

ADV_FSP.1.1D The developer shall provide a functional specification.

Content and presentation of evidence elements

ADV_FSP.1.1C The functional specification shall describe the TSF and its external interfaces using an informal style.

ADV_FSP.1.2C The functional specification shall describe the purpose and method of use of all external TSF interfaces, providing details of effects, exceptions and error messages, as appropriate.

ADV_FSP.1.3C The functional specification shall completely represent the TSF.

Evaluator action elements

ADV_FSP.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_FSP.1.2E The evaluator shall determine that the functional specification is an accurate and complete instantiation of the SFRs.

ADV_FSP.1.3E The evaluator shall determine that the functional specification is consistent with the TOE summary specification.

ADV_FSP.2 Fully defined external interfaces

Dependencies

ADV_RCR.1 Informal correspondence demonstration

Developer action elements

ADV_FSP.2.1D The developer shall provide a functional specification.

Content and presentation of evidence elements

ADV_FSP.2.1C The functional specification shall describe the TSF and its external interfaces using an informal style.

ADV_FSP.2.2C The functional specification shall describe the purpose and method of use of all external TSF interfaces, providing complete details of all effects, exceptions and error messages.

ADV_FSP.2.3C The functional specification shall completely represent the TSF.

ADV_FSP.2.4C The functional specification shall include rationale that the TSF is completely represented.

Evaluator action elements

ADV_FSP.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_FSP.2.2E The evaluator shall determine that the functional specification is an accurate and complete instantiation of the SFRs.

ADV_FSP.2.3E The evaluator shall determine that the functional specification is consistent with the TOE summary specification.

ADV_FSP.3 Semiformal functional specification

Dependencies

ADV_RCR.1 Informal correspondence demonstration

Developer action elements

ADV_FSP.3.1D The developer shall provide a functional specification.

Content and presentation of evidence elements

ADV_FSP.3.1C The functional specification shall describe the TSF and its external interfaces using a semiformal style, supported by informal, explanatory text where appropriate.

ADV_FSP.3.2C The functional specification shall describe the purpose and method of use of all external TSF interfaces, providing complete details of all effects, exceptions and error messages.

ADV_FSP.3.3C The functional specification shall completely represent the TSF.

ADV_FSP.3.4C The functional specification shall include rationale that the TSF is completely represented.

Evaluator action elements

ADV_FSP.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_FSP.3.2E The evaluator shall determine that the functional specification is an accurate and complete instantiation of the SFRs.

ADV_FSP.3.3E The evaluator shall determine that the functional specification is consistent with the TOE summary specification.

ADV_FSP.4 Formal functional specification

Dependencies

ADV_RCR.1 Informal correspondence demonstration

Developer action elements

ADV_FSP.4.1D The developer shall provide a functional specification.

Content and presentation of evidence elements

ADV_FSP.4.1C The functional specification shall describe the TSF and its external interfaces using a formal style, supported by informal, explanatory text where appropriate.

ADV_FSP.4.2C The functional specification shall describe the purpose and method of use of all external TSF interfaces, providing complete details of all effects, exceptions and error messages.

ADV_FSP.4.3C The functional specification shall completely represent the TSF.

ADV_FSP.4.4C The functional specification shall include rationale that the TSF is completely represented.

Evaluator action elements

ADV_FSP.4.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_FSP.4.2E The evaluator shall determine that the functional specification is an accurate and complete instantiation of the SFRs.

ADV_FSP.4.3E The evaluator shall determine that the functional specification is consistent with the TOE summary specification.

8.2 High-level design (ADV_HLD)

Objectives

278 The high-level design of a TOE provides a description of the TSF in terms of major structural units (i.e. subsystems) and relates these units to the functions that they provide. The high-level design requirements are intended to provide assurance that the TOE provides an architecture appropriate to implement the SFRs.

279 The high-level design refines the functional specification into subsystems. For each subsystem of the TSF, the high-level design describes its purpose

and function, and identifies the security functionality contained in the subsystem. The interrelationships of all subsystems are also defined in the high-level design. These interrelationships will be represented as external interfaces for data flow, control flow, etc., as appropriate.

Component levelling

280 The components in this family are levelled on the basis of the degree of formalism required of the high-level design, and on the degree of detail required for the interface specifications.

Application notes

281 The developer is expected to describe the design of the TSF in terms of subsystems. The term “subsystem” is used here to express the idea of decomposing the TSF into a relatively small number of parts. While the developer is not required to actually have “subsystems”, the developer is expected to represent a similar level of decomposition. For example, a design may be similarly decomposed using “layers”, “domains”, or “servers”.

282 The term “TSP-enforcing subsystem” refers to a subsystem that contributes to the enforcement of the TSP, either directly or indirectly.

283 The High-level design (ADV_HLD).*.2E elements within this family define a requirement that the evaluator determine that the high-level design is an accurate and complete instantiation of the SFRs. This provides a direct correspondence between the TOE security functional requirements and the high-level design, in addition to the pairwise correspondences required by the Representation correspondence (ADV_RCR) family. It is expected that the evaluator will use the evidence provided in Representation correspondence (ADV_RCR) as an input to making this determination, and the requirement for completeness is intended to be relative to the level of abstraction of the high-level design.

284 ADV_HLD.3.7C introduces a requirement for a complete presentation for the interfaces to the subsystems. This will provide the necessary detail for supporting both thorough testing of the TOE (using components from Depth (ATE_DPT)), and the assessment of vulnerabilities.

285 In the context of the level of formality of the high-level design, informal, semiformal and formal are considered to be hierarchical in nature. Thus, ADV_HLD.1.1C and ADV_HLD.2.1C may also be met with either a semiformal or formal high-level design, and ADV_HLD.3.1C and ADV_HLD.4.1C may also be met with a formal high-level design.

286 In High-level design (ADV_HLD).*.5C the phrase “underlying hardware, firmware and/or software” concerns the virtual machine on which the TOE runs (if any), rather than mechanisms contained within the TOE (which are covered elsewhere in the component). As such it is a requirement on information about the operational environment.

ADV_HLD.1 Descriptive high-level design

Dependencies

ADV_FSP.1 Informal functional specification
ADV_RCR.1 Informal correspondence demonstration

Developer action elements

ADV_HLD.1.1D The developer shall provide the high-level design of the TSF.

Content and presentation of evidence elements

ADV_HLD.1.1C The presentation of the high-level design shall be informal.

ADV_HLD.1.2C The high-level design shall describe the structure of the TSF in terms of subsystems.

ADV_HLD.1.3C The high-level design shall describe the security functionality provided by each subsystem of the TSF.

ADV_HLD.1.4C The high-level design shall identify any underlying hardware, firmware, and/or software required by the TSF with a presentation of the functions provided by the supporting protection mechanisms implemented in that hardware, firmware, or software.

ADV_HLD.1.5C The high-level design shall identify all interfaces to the subsystems of the TSF.

ADV_HLD.1.6C The high-level design shall identify which of the interfaces to the subsystems of the TSF are externally visible.

Evaluator action elements

ADV_HLD.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_HLD.1.2E The evaluator shall determine that the high-level design is an accurate and complete instantiation of the SFRs.

ADV_HLD.2 Security enforcing high-level design

Dependencies

ADV_FSP.1 Informal functional specification
ADV_RCR.1 Informal correspondence demonstration

Developer action elements

ADV_HLD.2.1D The developer shall provide the high-level design of the TSF.

Content and presentation of evidence elements

ADV_HLD.2.1C The presentation of the high-level design shall be informal.

ADV_HLD.2.2C The high-level design shall describe the structure of the TSF in terms of subsystems.

ADV_HLD.2.3C The high-level design shall describe the security functionality provided by each subsystem of the TSF.

ADV_HLD.2.4C The high-level design shall identify any underlying hardware, firmware, and/or software required by the TSF with a presentation of the functions provided by the supporting protection mechanisms implemented in that hardware, firmware, or software.

ADV_HLD.2.5C The high-level design shall identify all interfaces to the subsystems of the TSF.

ADV_HLD.2.6C The high-level design shall identify which of the interfaces to the subsystems of the TSF are externally visible.

ADV_HLD.2.7C The high-level design shall describe the purpose and method of use of all interfaces to the subsystems of the TSF, providing details of effects, exceptions and error messages, as appropriate.

ADV_HLD.2.8C The high-level design shall describe the separation of the TOE into TSP-enforcing and other subsystems.

Evaluator action elements

ADV_HLD.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_HLD.2.2E The evaluator shall determine that the high-level design is an accurate and complete instantiation of the SFRs.

ADV_HLD.3 Semiformal high-level design

Dependencies

ADV_FSP.3 Semiformal functional specification
ADV_RCR.2 Semiformal correspondence demonstration

Developer action elements

ADV_HLD.3.1D The developer shall provide the high-level design of the TSF.

Content and presentation of evidence elements

ADV_HLD.3.1C The presentation of the high-level design shall be semiformal.

ADV_HLD.3.2C The high-level design shall describe the structure of the TSF in terms of subsystems.

ADV_HLD.3.3C The high-level design shall describe the security functionality provided by each subsystem of the TSF.

ADV_HLD.3.4C The high-level design shall identify any underlying hardware, firmware, and/or software required by the TSF with a presentation of the functions provided by the supporting protection mechanisms implemented in that hardware, firmware, or software.

ADV_HLD.3.5C The high-level design shall identify all interfaces to the subsystems of the TSF.

ADV_HLD.3.6C The high-level design shall identify which of the interfaces to the subsystems of the TSF are externally visible.

ADV_HLD.3.7C The high-level design shall describe the purpose and method of use of all interfaces to the subsystems of the TSF, providing complete details of all effects, exceptions and error messages.

ADV_HLD.3.8C The high-level design shall describe the separation of the TOE into TSP-enforcing and other subsystems.

Evaluator action elements

ADV_HLD.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_HLD.3.2E The evaluator shall determine that the high-level design is an accurate and complete instantiation of the SFRs.

ADV_HLD.4 Semiformal high-level explanation

Dependencies

ADV_FSP.3 Semiformal functional specification
ADV_RCR.2 Semiformal correspondence demonstration

Developer action elements

ADV_HLD.4.1D The developer shall provide the high-level design of the TSF.

Content and presentation of evidence elements

ADV_HLD.4.1C The presentation of the high-level design shall be semiformal.

ADV_HLD.4.2C The high-level design shall describe the structure of the TSF in terms of subsystems.

ADV_HLD.4.3C The high-level design shall describe the security functionality provided by each subsystem of the TSF.

ADV_HLD.4.4C The high-level design shall identify any underlying hardware, firmware, and/or software required by the TSF with a presentation of the functions provided by the supporting protection mechanisms implemented in that hardware, firmware, or software.

ADV_HLD.4.5C The high-level design shall identify all interfaces to the subsystems of the TSF.

ADV_HLD.4.6C The high-level design shall identify which of the interfaces to the subsystems of the TSF are externally visible.

ADV_HLD.4.7C The high-level design shall describe the purpose and method of use of all interfaces to the subsystems of the TSF, providing complete details of all effects, exceptions and error messages.

ADV_HLD.4.8C The high-level design shall describe the separation of the TOE into TSP-enforcing and other subsystems.

Evaluator action elements

ADV_HLD.4.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_HLD.4.2E The evaluator shall determine that the high-level design is an accurate and complete instantiation of the SFRs.

ADV_HLD.5 Formal high-level design

Dependencies

ADV_FSP.4 Formal functional specification
ADV_RCR.3 Formal correspondence demonstration

Developer action elements

ADV_HLD.5.1D The developer shall provide the high-level design of the TSF.

Content and presentation of evidence elements

ADV_HLD.5.1C The presentation of the high-level design shall be formal.

ADV_HLD.5.2C The high-level design shall describe the structure of the TSF in terms of subsystems.

ADV_HLD.5.3C The high-level design shall describe the security functionality provided by each subsystem of the TSF.

ADV_HLD.5.4C The high-level design shall identify any underlying hardware, firmware, and/or software required by the TSF with a presentation of the functions provided by the supporting protection mechanisms implemented in that hardware, firmware, or software.

ADV_HLD.5.5C The high-level design shall identify all interfaces to the subsystems of the TSF.

ADV_HLD.5.6C The high-level design shall identify which of the interfaces to the subsystems of the TSF are externally visible.

ADV_HLD.5.7C The high-level design shall describe the purpose and method of use of all interfaces to the subsystems of the TSF, providing complete details of all effects, exceptions and error messages.

ADV_HLD.5.8C The high-level design shall describe the separation of the TOE into TSP-enforcing and other subsystems.

Evaluator action elements

ADV_HLD.5.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_HLD.5.2E The evaluator shall determine that the high-level design is an accurate and complete instantiation of the SFRs.

8.3 Implementation representation (ADV_IMP)

Objectives

287 The description of the implementation representation in the form of source code, firmware, hardware drawings, etc. captures the detailed internal workings of the TSF in support of analysis.

Component levelling

288 The components in this family are levelled on the basis of the completeness and structure of the implementation representation provided.

Application notes

289 The implementation representation is used to express the notion of the least abstract representation of the TSF, specifically the one that is used to create the TSF itself without further design refinement. Source code that is then compiled or a hardware drawing that is used to build the actual hardware are examples of parts of an implementation representation.

290 It is possible that evaluators may use the implementation representation to directly support other evaluation activities (e.g. vulnerability analysis, test coverage analysis, or identification of additional evaluator tests). It is

expected that PP/ST authors will select a component that requires that the implementation is complete and comprehensive enough to address the needs of all other SARs included in the PP/ST.

ADV_IMP.1 Subset of the implementation of the TSF

Dependencies

ADV_LLD.1 Descriptive low-level design
ADV_RCR.1 Informal correspondence demonstration
ALC_TAT.1 Well-defined development tools

Application notes

291 ADV_IMP.1.1D requires that the developer provide the implementation representation for a subset of the TSF. The intention is that access to at least a portion of the TSF will provide the evaluator with an opportunity to examine the implementation representation for those portions of the TSF where such an examination can add significantly to the understanding of, and assurance in, the mechanisms employed. Provision of a sample of the implementation representation will also allow the evaluator to sample the traceability evidence to gain assurance in the approach taken for refinement, and to assess the presentation of the implementation representation itself.

292 ADV_IMP.1.2E element defines a requirement that the evaluator determine that the least abstract TSF representation is an accurate and complete instantiation of the SFRs. This provides a direct correspondence between the SFRs and the least abstract TSF representation, in addition to the pairwise correspondences required by the Representation correspondence (ADV_RCR) family. It is expected that the evaluator will use the evidence provided in Representation correspondence (ADV_RCR) as an input to making this determination. The least abstract TSF representation for this component is an aggregate of the implementation representation that is provided and that portion of the low-level design for which no corresponding implementation representation is provided.

Developer action elements

ADV_IMP.1.1D The developer shall provide the implementation representation for a selected subset of the TSF.

Content and presentation of evidence elements

ADV_IMP.1.1C The implementation representation shall unambiguously define the TSF to a level of detail such that the TSF can be generated without further design decisions.

Evaluator action elements

ADV_IMP.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_IMP.1.2E The evaluator shall determine that the least abstract TSF representation provided is an accurate and complete instantiation of the SFRs.

ADV_IMP.2 Implementation of the TSF

Dependencies

ADV_LLD.1 Descriptive low-level design
ALC_TAT.1 Well-defined development tools

Application notes

293 The ADV_IMP.2.2E element defines a requirement that the evaluator determine that the implementation representation is an accurate and complete instantiation of the SFRs. This provides a direct correspondence between the TOE security functional requirements and the implementation representation, in addition to the pairwise correspondences required by the Representation correspondence (ADV_RCR) family. It is expected that the evaluator will use the evidence provided in Representation correspondence (ADV_RCR) as an input to making this determination.

Developer action elements

ADV_IMP.2.1D The developer shall provide the implementation representation for the entire TSF.

Content and presentation of evidence elements

ADV_IMP.2.1C The implementation representation shall unambiguously define the TSF to a level of detail such that the TSF can be generated without further design decisions.

ADV_IMP.2.2C The implementation representation shall describe the relationships between all portions of the implementation.

Evaluator action elements

ADV_IMP.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_IMP.2.2E The evaluator shall determine that the implementation representation is an accurate and complete instantiation of the SFRs.

ADV_IMP.3 Structured implementation of the TSF

Dependencies

ADV_INT.1 Modularity

ADV_LLD.1 Descriptive low-level design
ADV_RCR.1 Informal correspondence demonstration
ALC_TAT.1 Well-defined development tools

Application notes

294 The ADV_IMP.3.2E element defines a requirement that the evaluator determine that the implementation representation is an accurate and complete instantiation of the SFRs. This provides a direct correspondence between the TOE security functional requirements and the implementation representation, in addition to the pairwise correspondences required by the Representation correspondence (ADV_RCR) family. It is expected that the evaluator will use the evidence provided in Representation correspondence (ADV_RCR) as an input to making this determination.

Developer action elements

ADV_IMP.3.1D The developer shall provide the implementation representation for the entire TSF.

Content and presentation of evidence elements

ADV_IMP.3.1C The implementation representation shall unambiguously define the TSF to a level of detail such that the TSF can be generated without further design decisions.

ADV_IMP.3.2C The implementation representation shall describe the relationships between all portions of the implementation.

ADV_IMP.3.3C The implementation representation shall be structured into small and comprehensible sections.

Evaluator action elements

ADV_IMP.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_IMP.3.2E The evaluator shall determine that the implementation representation is an accurate and complete instantiation of the SFRs.

8.4 TSF internals (ADV_INT)

Objectives

295 This family addresses the internal structure of the TSF. Requirements are presented for modularity, layering (to separate levels of abstraction and minimise circular dependencies), minimisation of the complexity of policy enforcement mechanisms, and the minimisation of the amount of non-TSP-

enforcing functionality within the TSF -- thus resulting in a TSF that is simple enough to be analysed.

296 Modular design reduces the interdependence between elements of the TSF and thus reduces the risk that a change or error in one module will have effects throughout the TOE. Thus, a modular design provides the basis for determining the scope of interaction with other elements of the TSF, provides for increased assurance that unexpected effects do not occur, and also provides the basis for designing and evaluating test suites.

297 The use of layering and of simpler designs for the TSP-enforcing functionality reduces the complexity of the TSF. This in turn enables a better understanding of the TSF, providing more assurance that the SFRs are accurately and completely instantiated in the implementation.

298 Minimising the amount of functionality in the TSF that does not enforce the TSP, reduces the possibility of flaws in the TSF. In combination with modularity and layering, it allows the evaluator to focus only on that functionality which is necessary for TSP enforcement.

299 Design complexity minimisation contributes to the assurance that the code is understood -- the less complex the code in the TSF, the greater the likelihood that the design of the TSF is comprehensible. Design complexity minimisation is a key characteristic of a reference validation mechanism.

Component levelling

300 The components in this family are levelled on the basis of the amount of structure and minimisation required.

Application notes

301 The term “portions of the TSF” is used to represent parts of the TSF with a varying granularity based on the available TSF representations. The functional specification allows identification in terms of interfaces, the high-level design allows identification in terms of subsystems, the low-level design allows identification in terms of modules, and the implementation representation allows identification in terms of implementation units.

302 The ADV_INT.2.5C and ADV_INT.3.5C elements address minimisation of mutual interactions between layers. Nevertheless, it is still permissible to have mutual interactions between layers, but in such cases the developer is required to demonstrate that these mutual interactions are necessary and cannot reasonably be avoided.

303 ADV_INT.2.6C introduces a reference monitor concept by requiring the minimisation of complexity of the portions of the TSF that enforce the access control and/or information flow control policies identified in the TSP. ADV_INT.3.6C further develops the reference monitor concept by requiring minimisation of the complexity of the entire TSF.

304 Several of the elements within the components for this family refer to the architectural description. The architectural description is at a similar level of abstraction to the low-level design, in that it is concerned with the modules of the TSF. Whereas the low-level design describes the design of the modules of the TSF, the purpose of the architectural description is to provide evidence of modularity, layering, and minimisation of complexity of the TSF, as applicable. Both the low-level design and the implementation representation are required to be in compliance with the architectural description, to provide assurance that these TSF representations possess the required modularity, layering, and minimisation of complexity.

ADV_INT.1 Modularity

Dependencies

ADV_IMP.1 Subset of the implementation of the TSF
ADV_LLD.1 Descriptive low-level design

Developer action elements

ADV_INT.1.1D The developer shall design and structure the TSF in a modular fashion that avoids unnecessary interactions between the modules of the design.

ADV_INT.1.2D The developer shall provide an architectural description.

Content and presentation of evidence elements

ADV_INT.1.1C The architectural description shall identify the modules of the TSF.

ADV_INT.1.2C The architectural description shall describe the purpose, interface, parameters, and effects of each module of the TSF.

ADV_INT.1.3C The architectural description shall describe how the TSF design provides for largely independent modules that avoid unnecessary interactions.

Evaluator action elements

ADV_INT.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_INT.1.2E The evaluator shall determine that both the low-level design and the implementation representation are in compliance with the architectural description.

ADV_INT.2 Reduction of complexity

Dependencies

ADV_IMP.1 Subset of the implementation of the TSF
ADV_LLD.1 Descriptive low-level design

Application notes

305 This component introduces a reference monitor concept by requiring the minimisation of complexity of the portions of the TSF that enforce the access control and/or information flow control policies identified in the TSP.

Developer action elements

ADV_INT.2.1D The developer shall design and structure the TSF in a modular fashion that avoids unnecessary interactions between the modules of the design.

ADV_INT.2.2D The developer shall provide an architectural description.

ADV_INT.2.3D The developer shall design and structure the TSF in a layered fashion that minimises mutual interactions between the layers of the design.

ADV_INT.2.4D The developer shall design and structure the TSF in such a way that minimises the complexity of the portions of the TSF that enforce any access control and/or information flow control policies.

Content and presentation of evidence elements

ADV_INT.2.1C The architectural description shall identify the modules of the TSF and shall specify which portions of the TSF enforce the access control and/or information flow control policies.

ADV_INT.2.2C The architectural description shall describe the purpose, interface, parameters, and effects of each module of the TSF.

ADV_INT.2.3C The architectural description shall describe how the TSF design provides for largely independent modules that avoid unnecessary interactions.

ADV_INT.2.4C The architectural description shall describe the layering architecture.

ADV_INT.2.5C The architectural description shall show that mutual interactions have been minimised, and justify those that remain.

ADV_INT.2.6C The architectural description shall describe how the portions of the TSF that enforce any access control and/or information flow control policies have been structured to minimise complexity.

Evaluator action elements

ADV_INT.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_INT.2.2E The evaluator shall determine that both the low-level design and the implementation representation are in compliance with the architectural description.

ADV_INT.3 Minimisation of complexity

Dependencies

ADV_IMP.2 Implementation of the TSF
ADV_LLD.1 Descriptive low-level design

Application notes

306 This component requires that the reference monitor property "simple enough to be analysed" is fully addressed. When this component is combined with the SFRs FPT_RVM.1 and FPT_SEP.3, the reference monitor concept would be fully realised.

Developer action elements

ADV_INT.3.1D The developer shall design and structure the TSF in a modular fashion that avoids unnecessary interactions between the modules of the design.

ADV_INT.3.2D The developer shall provide an architectural description.

ADV_INT.3.3D The developer shall design and structure the TSF in a layered fashion that minimises mutual interactions between the layers of the design.

ADV_INT.3.4D The developer shall design and structure the TSF in such a way that minimises the complexity of the entire TSF.

ADV_INT.3.5D The developer shall design and structure the portions of the TSF that enforce any access control and/or information flow control policies such that they are simple enough to be analysed.

ADV_INT.3.6D The developer shall ensure that functions whose objectives are not relevant for the TSF are excluded from the TSF modules.

Content and presentation of evidence elements

ADV_INT.3.1C The architectural description shall identify the modules of the TSF and shall specify which portions of the TSF enforce the access control and/or information flow control policies.

ADV_INT.3.2C The architectural description shall describe the purpose, interface, parameters, and side-effects of each module of the TSF.

ADV_INT.3.3C The architectural description shall describe how the TSF design provides for largely independent modules that avoid unnecessary interactions.

ADV_INT.3.4C The architectural description shall describe the layering architecture.

ADV_INT.3.5C The architectural description shall show that mutual interactions have been minimised, and justify those that remain.

ADV_INT.3.6C The architectural description shall describe how the entire TSF has been structured to minimise complexity.

ADV_INT.3.7C The architectural description shall justify the inclusion of any non-TSP-enforcing modules in the TSF.

Evaluator action elements

ADV_INT.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_INT.3.2E The evaluator shall determine that both the low-level design and the implementation representation are in compliance with the architectural description.

ADV_INT.3.3E The evaluator shall confirm that the portions of the TSF that enforce any access control and/or information flow control policies are simple enough to be analysed.

8.5 Low-level design (ADV_LLD)

Objectives

307 The low-level design of a TOE provides a description of the internal workings of the TSF in terms of modules and their interrelationships and dependencies. The low-level design provides assurance that the TSF subsystems have been correctly and effectively refined.

308 For each module of the TSF, the low-level design describes its purpose, function, interfaces, dependencies, and the implementation of any TSP-enforcing functions.

Component levelling

309 The components in this family are levelled on the basis of the degree of formalism required of the low-level design, and on the degree of detail required for the interface specifications.

Application notes

310 The term “TSP-enforcing module” refers to any module that must be relied upon for correct enforcement of the TSP.

311 The term “security functionality” is used to represent the set of operations that a module performs in contribution to security functions implemented by the TOE. This distinction is made because modules do not necessarily relate to specific security functions. While a given module may correspond directly to a security function, or even multiple security functions, it is also possible that many modules must be combined to implement a single security function.

312 The Low-level design (ADV_LLD).*.6C elements require that the low-level design describe how each TSP-enforcing function is provided. The intent of this requirement is that the low-level design provide a description of how each module is expected to be implemented from a design perspective.

313 The Low-level design (ADV_LLD).*.2E elements within this family define a requirement that the evaluator determine that the low-level design is an accurate and complete instantiation of the SFRs. This provides a direct correspondence between the SFRs and the low-level design, in addition to the pairwise correspondences required by the Representation correspondence (ADV_RCR) family. It is expected that the evaluator will use the evidence provided in Representation correspondence (ADV_RCR) as an input to making this determination, and the requirement for completeness is intended to be relative to the level of abstraction of the low-level design.

314 ADV_LLD.2.8C introduces a requirement for a complete presentation for the interfaces to the modules. This will provide the necessary detail for supporting both thorough testing of the TOE (using components from Depth (ATE_DPT)), and the assessment of vulnerabilities.

315 In the context of the level of formality of the low-level design, informal, semiformal and formal are considered to be hierarchical in nature. Thus, ADV_LLD.1.1C may also be met with either a semiformal or formal low-level design, and ADV_LLD.2.1C may also be met with a formal low-level design.

ADV_LLD.1 Descriptive low-level design

Dependencies

ADV_HLD.2 Security enforcing high-level design
ADV_RCR.1 Informal correspondence demonstration

Developer action elements

ADV_LLD.1.1D The developer shall provide the low-level design of the TSF.

Content and presentation of evidence elements

ADV_LLD.1.1C The presentation of the low-level design shall be informal.

ADV_LLD.1.2C The low-level design shall describe the TSF in terms of modules.

ADV_LLD.1.3C The low-level design shall describe the purpose of each module.

ADV_LLD.1.4C The low-level design shall define the interrelationships between the modules in terms of provided security functionality and dependencies on other modules.

ADV_LLD.1.5C The low-level design shall describe how each TSP-enforcing module is provided.

ADV_LLD.1.6C The low-level design shall identify all interfaces to the modules of the TSF.

ADV_LLD.1.7C The low-level design shall identify which of the interfaces to the modules of the TSF are externally visible.

ADV_LLD.1.8C The low-level design shall describe the purpose and method of use of all interfaces to the modules of the TSF, providing details of effects, exceptions and error messages, as appropriate.

ADV_LLD.1.9C The low-level design shall describe the separation of the TOE into TSP-enforcing and other modules.

Evaluator action elements

ADV_LLD.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_LLD.1.2E The evaluator shall determine that the low-level design is an accurate and complete instantiation of the SFRs.

ADV_LLD.2 Semiformal low-level design

Dependencies

ADV_HLD.3 Semiformal high-level design
ADV_RCR.2 Semiformal correspondence demonstration

Developer action elements

ADV_LLD.2.1D The developer shall provide the low-level design of the TSF.

Content and presentation of evidence elements

ADV_LLD.2.1C The presentation of the low-level design shall be semiformal.

ADV_LLD.2.2C The low-level design shall describe the TSF in terms of modules.

ADV_LLD.2.3C The low-level design shall describe the purpose of each module.

ADV_LLD.2.4C The low-level design shall define the interrelationships between the modules in terms of provided security functionality and dependencies on other modules.

ADV_LLD.2.5C The low-level design shall describe how each TSP-enforcing module is provided.

ADV_LLD.2.6C The low-level design shall identify all interfaces to the modules of the TSF.

ADV_LLD.2.7C The low-level design shall identify which of the interfaces to the modules of the TSF are externally visible.

ADV_LLD.2.8C The low-level design shall describe the purpose and method of use of all interfaces to the modules of the TSF, providing complete details of all effects, exceptions and error messages.

ADV_LLD.2.9C The low-level design shall describe the separation of the TOE into TSP-enforcing and other modules.

Evaluator action elements

ADV_LLD.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_LLD.2.2E The evaluator shall determine that the low-level design is an accurate and complete instantiation of the SFRs.

ADV_LLD.3 Formal low-level design

Dependencies

ADV_HLD.5 Formal high-level design
ADV_RCR.3 Formal correspondence demonstration

Developer action elements

ADV_LLD.3.1D The developer shall provide the low-level design of the TSF.

Content and presentation of evidence elements

ADV_LLD.3.1C The presentation of the low-level design shall be formal.

ADV_LLD.3.2C The low-level design shall describe the TSF in terms of modules.

ADV_LLD.3.3C The low-level design shall describe the purpose of each module.

ADV_LLD.3.4C The low-level design shall define the interrelationships between the modules in terms of provided security functionality and dependencies on other modules.

ADV_LLD.3.5C The low-level design shall describe how each TSP-enforcing module is provided.

ADV_LLD.3.6C The low-level design shall identify all interfaces to the modules of the TSF.

ADV_LLD.3.7C The low-level design shall identify which of the interfaces to the modules of the TSF are externally visible.

ADV_LLD.3.8C The low-level design shall describe the purpose and method of use of all interfaces to the modules of the TSF, providing complete details of all effects, exceptions and error messages.

ADV_LLD.3.9C The low-level design shall describe the separation of the TOE into TSP-enforcing and other modules.

Evaluator action elements

ADV_LLD.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_LLD.3.2E The evaluator shall determine that the low-level design is an accurate and complete instantiation of the SFRs.

8.6 Representation correspondence (ADV_RCR)

Objectives

316 The correspondence between the SFRs and the functional specification, and the pairwise correspondence between the various TSF representations (i.e. functional specification, high-level design, low-level design, implementation representation) together address the correct and complete instantiation of the SFRs to the least abstract TSF representation provided. This conclusion is achieved by step-wise refinement and the cumulative results of correspondence determinations between all adjacent abstractions.

Component levelling

317 The components in this family are levelled on the basis of the required level of formality of the correspondence between the various TSF representations.

Application notes

318 The developer must demonstrate to the evaluator that the most detailed, or least abstract, TSF representation provided is an accurate, consistent, and complete instantiation of the SFRs. This is accomplished by showing correspondence between adjacent representations at a commensurate level of rigour.

319 This family of requirements is not intended to address correspondence relating to the TSP model or the TSP. Rather, as shown in Figure **9**, it is intended to address correspondence between various TSF representations (i.e. the SFRs, functional specification, high-level design, low-level design, and implementation representation) that are provided.

320 The Representation correspondence (ADV_RCR).*.1C elements refer to “all relevant security functionality” in defining the scope of what must be refined between an adjacent pair of TSF representations. Where the implementation representation is only provided for a subset of the TSF (as in ADV_IMP.1 Subset of the implementation of the TSF), the required refinements between the low-level design and the implementation representation are limited to the security functionality that is presented in the implementation representation. In all other cases, this element requires that all parts of the

more abstract TSF representation be refined in the less abstract TSF representation.

321 In the context of the level of formality for correspondence between adjacent TSF representations, informal, semiformal and formal are considered to be hierarchical in nature. Thus, ADV_RCR.2.2C and ADV_RCR.3.2C may be met with a formal proof of correspondence, and in the absence of any requirements on its level of formality, a demonstration of correspondence may be informal, semiformal or formal.

ADV_RCR.1 Informal correspondence demonstration

Developer action elements

ADV_RCR.1.1D The developer shall provide an analysis of correspondence between all adjacent pairs of TSF representations that are provided.

Content and presentation of evidence elements

ADV_RCR.1.1C For each adjacent pair of provided TSF representations, the analysis shall demonstrate that all relevant security functionality of the more abstract TSF representation is correctly and completely refined in the less abstract TSF representation.

Evaluator action elements

ADV_RCR.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_RCR.2 Semiformal correspondence demonstration

Developer action elements

ADV_RCR.2.1D The developer shall provide an analysis of correspondence between all adjacent pairs of TSF representations that are provided.

Content and presentation of evidence elements

ADV_RCR.2.1C For each adjacent pair of provided TSF representations, the analysis shall demonstrate that all relevant security functionality of the more abstract TSF representation is correctly and completely refined in the less abstract TSF representation.

ADV_RCR.2.2C For each adjacent pair of provided TSF representations, where portions of both representations are at least semiformally specified, the demonstration of correspondence between those portions of the representations shall be semiformal.

Evaluator action elements

ADV_RCR.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_RCR.3 Formal correspondence demonstration

Application notes

322 The developer must either demonstrate or prove correspondence, as described in the requirements below, commensurate with the level of rigour of presentation style. For example, correspondence must be proven when corresponding representations are formally specified.

Developer action elements

ADV_RCR.3.1D The developer shall provide an analysis of correspondence between all adjacent pairs of TSF representations that are provided.

Content and presentation of evidence elements

ADV_RCR.3.1C For those corresponding portions of representations that are formally specified, the developer shall prove that correspondence.

ADV_RCR.3.2C For each adjacent pair of provided TSF representations, the analysis shall prove or demonstrate that all relevant security functionality of the more abstract TSF representation is correctly and completely refined in the less abstract TSF representation.

ADV_RCR.3.3C For each adjacent pair of provided TSF representations, where portions of one representation are semiformally specified and the other at least semiformally specified, the demonstration of correspondence between those portions of the representations shall be semiformal .

ADV_RCR.3.4C For each adjacent pair of provided TSF representations, where portions of both representations are formally specified, the proof of correspondence between those portions of the representations shall be formal.

Evaluator action elements

ADV_RCR.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_RCR.3.2E The evaluator shall determine the accuracy of the proofs of correspondence by selectively verifying the formal analysis.

8.7 Security policy modeling (ADV_SPM)

Objectives

323 It is the objective of this family to provide additional assurance that the security functions in the functional specification enforce the policies in the TSP. This is accomplished via the development of a security policy model that is based on a subset of the policies of the TSP, and establishing a correspondence between the functional specification, the security policy model, and these policies of the TSP.

Component levelling

324 The components in this family are levelled on the basis of the degree of formality required of the TSP model, and the degree of formality required of the correspondence between the TSP model and the functional specification.

Application notes

325 While a TSP may include any policies, TSP models have traditionally represented only subsets of those policies, because modeling certain policies is currently beyond the state of the art. The current state of the art determines the policies that can be modeled, and the PP/ST author should identify specific functions and associated policies that can, and thus are required to be, modeled. At the very least, access control and information flow control policies are required to be modeled (if they are part of the TSP) since they are within the state of the art.

326 For each of the components within this family, there is a requirement to describe the rules and characteristics of applicable policies of the TSP in the TSP model and to ensure that the TSP model satisfies the corresponding policies of the TSP. The “rules” and “characteristics” of a TSP model are intended to allow flexibility in the type of model that may be developed (e.g. state transition, non-interference). For example, rules may be represented as “properties” (e.g. simple security property) and characteristics may be represented as definitions such as “initial state”, “secure state”, “subjects” and “objects”.

327 In the context of the level of formality of the TSP model and the correspondence between the TSP model and the functional specification, informal, semiformal and formal are considered to be hierarchical in nature. Thus, ADV_SPM.1.1C may also be met with either a semiformal or formal TSP model, and ADV_SPM.2.1C may also be met with a formal TSP model. Furthermore, ADV_SPM.2.5C and ADV_SPM.3.5C may be met with a formal proof of correspondence. Finally, in the absence of any requirements on its level of formality, a demonstration of correspondence may be informal, semiformal or formal.

ADV_SPM.1 Informal TOE security policy model

Dependencies

ADV_FSP.1 Informal functional specification

Developer action elements

ADV_SPM.1.1D The developer shall provide a TSP model.

ADV_SPM.1.2D The developer shall demonstrate correspondence between the functional specification and the TSP model.

Content and presentation of evidence elements

ADV_SPM.1.1C The TSP model shall be informal.

ADV_SPM.1.2C The TSP model shall describe the rules and characteristics of all policies of the TSP that can be modeled.

ADV_SPM.1.3C The TSP model shall include a rationale that demonstrates that it is consistent and complete with respect to all policies of the TSP that can be modeled.

ADV_SPM.1.4C The demonstration of correspondence between the TSP model and the functional specification shall show that all of the external interfaces to the TSF in the functional specification are consistent and complete with respect to the TSP model.

Evaluator action elements

ADV_SPM.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_SPM.2 Semiformal TOE security policy model

Dependencies

ADV_FSP.1 Informal functional specification

Developer action elements

ADV_SPM.2.1D The developer shall provide a TSP model.

ADV_SPM.2.2D The developer shall demonstrate correspondence between the functional specification and the TSP model.

Content and presentation of evidence elements

ADV_SPM.2.1C The TSP model shall be semiformal.

ADV_SPM.2.2C The TSP model shall describe the rules and characteristics of all policies of the TSP that can be modeled.

ADV_SPM.2.3C The TSP model shall include a rationale that demonstrates that it is consistent and complete with respect to all policies of the TSP that can be modeled.

ADV_SPM.2.4C The demonstration of correspondence between the TSP model and the functional specification shall show that all of the external interfaces in the functional specification are consistent and complete with respect to the TSP model.

ADV_SPM.2.5C Where the functional specification is at least semiformal, the demonstration of correspondence between the TSP model and the functional specification shall be semiformal.

Evaluator action elements

ADV_SPM.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_SPM.3 Formal TOE security policy model

Dependencies

ADV_FSP.1 Informal functional specification

Developer action elements

ADV_SPM.3.1D The developer shall provide a TSP model.

ADV_SPM.3.2D The developer shall demonstrate or prove, as appropriate, correspondence between the functional specification and the TSP model.

Content and presentation of evidence elements

ADV_SPM.3.1C The TSP model shall be formal.

ADV_SPM.3.2C The TSP model shall describe the rules and characteristics of all policies of the TSP that can be modeled.

ADV_SPM.3.3C The TSP model shall include a rationale that demonstrates that it is consistent and complete with respect to all policies of the TSP that can be modeled.

ADV_SPM.3.4C The demonstration of correspondence between the TSP model and the functional specification shall show that all of the extrernal interfaces in the functional specification are consistent and complete with respect to the TSP model.

ADV_SPM.3.5C Where the functional specification is semiformal, the demonstration of correspondence between the TSP model and the functional specification shall be semiformal.

ADV_SPM.3.6C Where the functional specification is formal, the proof of correspondence between the TSP model and the functional specification shall be formal.

Evaluator action elements

ADV_SPM.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

9 Class AGD: Guidance documents

328 The guidance documents class provides the requirements for user and administrator guidance documentation. For the secure administration and use of the TOE it is necessary to describe all relevant aspects for the secure application of the TOE. Guidance documentation includes user and administrator guidance and, when included in the assurance requirements, the specific guidance for users and administrators resulting from the requirements in the ADO: Delivery and operation class and the Flaw remediation (ALC_FLR) family.

329 Figure **11** shows the families within this class, and the hierarchy of components within the families.

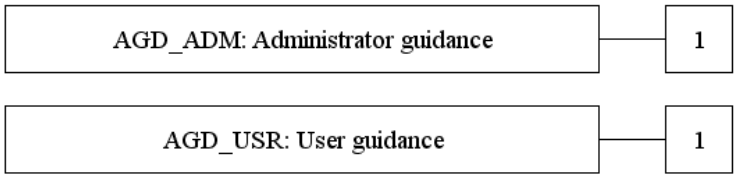

Figure 11 - AGD: Guidance documents class decomposition

9.1 Administrator guidance (AGD_ADM)

Objectives

330 Administrator guidance refers to written material that is intended to be used by those persons responsible for configuring, maintaining, and administering the TOE in a correct manner for maximum security. Because the secure operation of the TOE is dependent upon the correct performance of the TSF, persons responsible for performing these functions are trusted by the TSF. Administrator guidance is intended to help administrators understand the TSF, including the security-critical information provided by the TSF, and the security-critical actions required by the administrator.

Component levelling

331 This family contains only one component.

Application notes

332 The requirements AGD_ADM.1.3C and encompass the aspect that any warnings to the users of a TOE with regard to the TOE security environment and the security objectives described in the PP/ST are appropriately covered in the administrator guidance.

333 The concept of secure values, as employed in AGD_ADM.1.4C, has relevance where an administrator has control over security parameters. Guidance needs to be provided on secure and insecure settings for such parameters. This concept is related to the use of the component FMT_MSA.2 from CC Part 2.

334 AGD_ADM.1.5C requires that the administrator guidance describe the appropriate administrator's reactions to all security-relevant events. Although many security-relevant events are the result of performing administrative functions, this need not always be the case (e.g. the audit log fills up, an intrusion is detected). Furthermore, a security-relevant event may happen as a result of a specific chain of administrator functions or, conversely, several security-relevant events may be triggered by one function.

AGD_ADM.1 Administrator guidance

Dependencies

ADV_FSP.1 Informal functional specification

Developer action elements

AGD_ADM.1.1D The developer shall provide administrator guidance addressed to system administrative personnel.

Content and presentation of evidence elements

AGD_ADM.1.1C The administrator guidance shall describe the administrative functions and interfaces available to the administrator of the TOE.

AGD_ADM.1.2C The administrator guidance shall describe how to administer the TOE in a secure manner.

AGD_ADM.1.3C The administrator guidance shall contain warnings about functions and privileges that should be controlled in a secure processing environment.

AGD_ADM.1.4C The administrator guidance shall describe all security parameters under the control of the administrator, indicating secure values as appropriate.

AGD_ADM.1.5C The administrator guidance shall describe each type of security-relevant event relative to the administrative functions that need to be performed, including changing the security characteristics of entities under the control of the TSF.

AGD_ADM.1.6C The administrator guidance shall describe all security objectives for the operational environment that are relevant to the administrator.

Evaluator action elements

AGD_ADM.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

9.2 User guidance (AGD_USR)

Objectives

335 User guidance refers to material that is intended to be used by non-administrative human users of the TOE, and by others (e.g. programmers) using the TOE's external interfaces. User guidance describes the security functions provided by the TSF and provides instructions and guidelines, including warnings, for its secure use.

336 The user guidance provides a measure of confidence that non-malicious users, application providers and others exercising the external interfaces of the TOE will understand the secure operation of the TOE and will use it as intended.

Component levelling

337 This family contains only one component.

Application notes

338 In many cases it may be appropriate that guidance is provided in separate documents: one for human users, and one for application programmers and/or hard-ware designers using software or hardware interfaces.

AGD_USR.1 User guidance

Dependencies

ADV_FSP.1 Informal functional specification

Developer action elements

AGD_USR.1.1D The developer shall provide user guidance.

Content and presentation of evidence elements

AGD_USR.1.1C The user guidance shall describe the interfaces available to the non-administrative users of the TOE.

AGD_USR.1.2C The user guidance shall describe the use of the interfaces available to the non-administrative users of the TOE.

AGD_USR.1.3C The user guidance shall contain warnings about user-accessible functions and privileges that should be controlled in a secure processing environment.

AGD_USR.1.4C The user guidance shall describe all security objectives for the operational environment that are relevant to the user.

Evaluator action elements

AGD_USR.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

10 Class ALC: Life cycle support

339 Life-cycle support is an aspect of establishing discipline and control in the processes of refinement of the TOE during its development and maintenance. Confidence in the correspondence between the TOE security requirements and the TOE is greater if security analysis and the production of the evidence are done on a regular basis as an integral part of the development and maintenance activities.

340 Figure **12** shows the families within this class, and the hierarchy of components within the families.

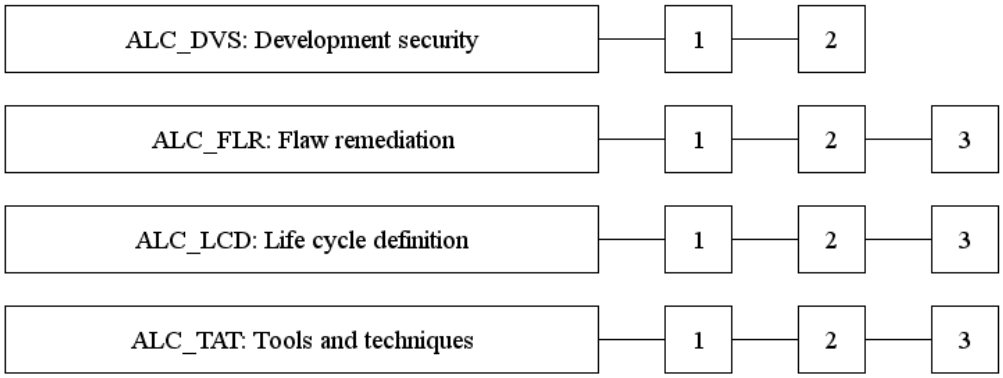

Figure 12 - ALC: Life cycle support class decomposition

10.1 Development security (ALC_DVS)

Objectives

341 Development security is concerned with physical, procedural, personnel, and other security measures that may be used in the development environment to protect the TOE. It includes the physical security of the development location and any procedures used to select development staff.

Component levelling

342 The components in this family are levelled on the basis of whether justification of the sufficiency of the security measures is required.

Application notes

343 This family deals with measures to remove or reduce threats to assets in the development environment of the TOE.

344 The evaluator should determine whether there is a need for visiting the developer's site in order to confirm that the requirements of this family are met.

345 It is recognised that confidentiality may not always be an issue for the protection of the TOE in its development environment. The use of the word "necessary" allows for the selection of appropriate safeguards.

ALC_DVS.1 Identification of security measures

Developer action elements

ALC_DVS.1.1D The developer shall produce development security documentation.

Content and presentation of evidence elements

ALC_DVS.1.1C The development security documentation shall describe all the physical, procedural, personnel, and other security measures that are necessary to protect the confidentiality and integrity of the TOE design and implementation in its development environment.

ALC_DVS.1.2C The development security documentation shall provide evidence that these security measures are followed during the development and maintenance of the TOE.

Evaluator action elements

ALC_DVS.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ALC_DVS.1.2E The evaluator shall confirm that the security measures are being applied.

ALC_DVS.2 Sufficiency of security measures

Developer action elements

ALC_DVS.2.1D The developer shall produce development security documentation.

Content and presentation of evidence elements

ALC_DVS.2.1C The development security documentation shall describe all the physical, procedural, personnel, and other security measures that are necessary to protect the confidentiality and integrity of the TOE design and implementation in its development environment.

ALC_DVS.2.2C The development security documentation shall provide evidence that these security measures are followed during the development and maintenance of the TOE.

ALC_DVS.2.3C The evidence shall justify that the security measures provide the necessary level of protection to maintain the confidentiality and integrity of the TOE.

Evaluator action elements

ALC_DVS.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ALC_DVS.2.2E The evaluator shall confirm that the security measures are being applied.

10.2 Flaw remediation (ALC_FLR)

Objectives

346 Flaw remediation requires that discovered security flaws be tracked and corrected by the developer. Although future compliance with flaw remediation procedures cannot be determined at the time of the TOE evaluation, it is possible to evaluate the policies and procedures that a developer has in place to track and correct flaws, and to distribute the flaw information and corrections.

Component levelling

347 The components in this family are levelled on the basis of the increasing extent in scope of the flaw remediation procedures and the rigour of the flaw remediation policies.

Application notes

348 This family provides assurance that the TOE will be maintained and supported in the future, requiring the TOE developer to track and correct flaws in the TOE. Additionally, requirements are included for the distribution of flaw corrections. However, this family does not impose evaluation requirements beyond the current evaluation.

349 The TOE user is considered to be the focal point in the user organisation that is responsible for receiving and implementing fixes to security flaws. This is not necessarily an individual user, but may be an organisational representative who is responsible for the handling of security flaws. The use of the term TOE user recognises that different organisations have different procedures for handling flaw reporting, which may be done either by an individual user, or by a central administrative body.

350 The flaw remediation procedures should describe the methods for dealing with all types of flaws encountered. These flaws may be reported by the developer, by users of the TOE, or by other parties with familiarity with the TOE. Some flaws may not be reparable immediately. There may be some occasions where a flaw cannot be fixed and other (e.g. procedural) measures must be taken. The documentation provided should cover the procedures for providing the operational sites with fixes, and providing information on flaws where fixes are delayed (and what to do in the interim) or when fixes are not possible.

351 Once the evaluation of a TOE is complete, it is no longer the target for evaluation. Furthermore, any changes to this evaluated TOE result in the original evaluation results being no longer applicable to the changed version. The phrase release of the TOE used in this family therefore refers to a version of a product or system that is a release of a certified TOE, to which changes have been applied.

ALC_FLR.1 Basic flaw remediation

Developer action elements

ALC_FLR.1.1D The developer shall provide flaw remediation procedures addressed to TOE developers.

Content and presentation of evidence elements

ALC_FLR.1.1C The flaw remediation procedures documentation shall describe the procedures used to track all reported security flaws in each release of the TOE.

ALC_FLR.1.2C The flaw remediation procedures shall require that a description of the nature and effect of each security flaw be provided, as well as the status of finding a correction to that flaw.

ALC_FLR.1.3C The flaw remediation procedures shall require that corrective actions be identified for each of the security flaws.

ALC_FLR.1.4C The flaw remediation procedures documentation shall describe the methods used to provide flaw information, corrections and guidance on corrective actions to TOE users.

Evaluator action elements

ALC_FLR.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ALC_FLR.2 Flaw reporting procedures

Objectives

352 In order for the developer to be able to act appropriately upon security flaw reports from TOE users, and to know to whom to send corrective fixes, TOE users need to understand how to submit security flaw reports to the developer. Flaw remediation guidance from the developer to the TOE user ensures that TOE users are aware of this important information.

Developer action elements

ALC_FLR.2.1D The developer shall provide flaw remediation procedures addressed to TOE developers.

ALC_FLR.2.2D The developer shall establish a procedure for accepting and acting upon all reports of security flaws and requests for corrections to those flaws.

ALC_FLR.2.3D The developer shall provide flaw remediation guidance addressed to TOE users.

Content and presentation of evidence elements

ALC_FLR.2.1C The flaw remediation procedures documentation shall describe the procedures used to track all reported security flaws in each release of the TOE.

ALC_FLR.2.2C The flaw remediation procedures shall require that a description of the nature and effect of each security flaw be provided, as well as the status of finding a correction to that flaw.

ALC_FLR.2.3C The flaw remediation procedures shall require that corrective actions be identified for each of the security flaws.

ALC_FLR.2.4C The flaw remediation procedures documentation shall describe the methods used to provide flaw information, corrections and guidance on corrective actions to TOE users.

ALC_FLR.2.5C The flaw remediation procedures shall describe a means by which the developer receives from TOE users reports and enquiries of suspected security flaws in the TOE.

ALC_FLR.2.6C The procedures for processing reported security flaws shall ensure that any reported flaws are corrected and the correction issued to TOE users.

ALC_FLR.2.7C The procedures for processing reported security flaws shall provide safeguards that any corrections to these security flaws do not introduce any new flaws.

ALC_FLR.2.8C The flaw remediation guidance shall describe a means by which TOE users report to the developer any suspected security flaws in the TOE.

Evaluator action elements

ALC_FLR.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ALC_FLR.3 Systematic flaw remediation

Objectives

353 In order for the developer to be able to act appropriately upon security flaw reports from TOE users, and to know to whom to send corrective fixes, TOE users need to understand how to submit security flaw reports to the developer, and how to register themselves with the developer so that they may receive these corrective fixes. Flaw remediation guidance from the developer to the TOE user ensures that TOE users are aware of this important information.

Developer action elements

ALC_FLR.3.1D The developer shall provide flaw remediation procedures addressed to TOE developers.

ALC_FLR.3.2D The developer shall establish a procedure for accepting and acting upon all reports of security flaws and requests for corrections to those flaws.

ALC_FLR.3.3D The developer shall provide flaw remediation guidance addressed to TOE users.

Content and presentation of evidence elements

ALC_FLR.3.1C The flaw remediation procedures documentation shall describe the procedures used to track all reported security flaws in each release of the TOE.

ALC_FLR.3.2C The flaw remediation procedures shall require that a description of the nature and effect of each security flaw be provided, as well as the status of finding a correction to that flaw.

ALC_FLR.3.3C The flaw remediation procedures shall require that corrective actions be identified for each of the security flaws.

ALC_FLR.3.4C The flaw remediation procedures documentation shall describe the methods used to provide flaw information, corrections and guidance on corrective actions to TOE users.

ALC_FLR.3.5C The flaw remediation procedures shall describe a means by which the developer receives from TOE users reports and enquiries of suspected security flaws in the TOE.

ALC_FLR.3.6C The procedures for processing reported security flaws shall ensure that any reported flaws are corrected and the correction issued to TOE users.

ALC_FLR.3.7C The procedures for processing reported security flaws shall provide safeguards that any corrections to these security flaws do not introduce any new flaws.

ALC_FLR.3.8C The flaw remediation guidance shall describe a means by which TOE users report to the developer any suspected security flaws in the TOE.

ALC_FLR.3.9C The flaw remediation procedures shall include a procedure requiring timely responses for the automatic distribution of security flaw reports and the associated corrections to registered users who might be affected by the security flaw.

ALC_FLR.3.10C The flaw remediation guidance shall describe a means by which TOE users may register with the developer, to be eligible to receive security flaw reports and corrections.

ALC_FLR.3.11C The flaw remediation guidance shall identify the specific points of contact for all reports and enquiries about security issues involving the TOE.

Evaluator action elements

ALC_FLR.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

10.3 Life cycle definition (ALC_LCD)

Objectives

354 Poorly controlled development and maintenance of the TOE can result in a TOE that does not meet all of its SFRs. Therefore, it is important that a model for the development and maintenance of a TOE be established as early as possible in the TOE's life-cycle.

355 Using a model for the development and maintenance of a TOE does not guarantee that the TOE meets all of its SFRs. It is possible that the model chosen will be insufficient or inadequate and therefore no benefits in the quality of the TOE can be observed. Using a life-cycle model that has been approved by some group of experts (e.g. academic experts, standards bodies) improves the chances that the development and maintenance models will contribute to the TOE meeting its SFRs.

Component levelling

356 The components in this family are levelled on the basis of increasing requirements for standardisation and measurability of the life-cycle model, and for compliance with that model.

Application notes

357 A life-cycle model encompasses the procedures, tools and techniques used to develop and maintain the TOE. Aspects of the process that may be covered by such a model include design methods, review procedures, project management controls, change control procedures, test methods and acceptance procedures. An effective life-cycle model will address these aspects of the development and maintenance process within an overall management structure that assigns responsibilities and monitors progress.

358 Although life-cycle definition deals with the maintenance of the TOE and hence with aspects becoming relevant after the completion of the evaluation, its evaluation adds assurance through an analysis of the life-cycle information for the TOE provided at the time of the evaluation.

359 A standardised life-cycle model is a model that has been approved by some group of experts (e.g. academic experts, standards bodies).

360 A measurable life-cycle model is a model with arithmetic parameters and/or metrics that measure TOE development properties (e.g. source code complexity metrics).

361 A life-cycle model provides for the necessary control over the development and maintenance of the TOE, if the developer can supply information that

shows that the model appropriately minimises the danger the TOE not meeting its SFRs.

ALC_LCD.1 Developer defined life-cycle model

Developer action elements

ALC_LCD.1.1D The developer shall establish a life-cycle model to be used in the development and maintenance of the TOE.

ALC_LCD.1.2D The developer shall provide life-cycle definition documentation.

Content and presentation of evidence elements

ALC_LCD.1.1C The life-cycle definition documentation shall describe the model used to develop and maintain the TOE.

ALC_LCD.1.2C The life-cycle model shall provide for the necessary control over the development and maintenance of the TOE.

Evaluator action elements

ALC_LCD.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ALC_LCD.2 Standardised life-cycle model

Developer action elements

ALC_LCD.2.1D The developer shall establish a life-cycle model to be used in the development and maintenance of the TOE.

ALC_LCD.2.2D The developer shall provide life-cycle definition documentation.

ALC_LCD.2.3D The developer shall use a standardised life-cycle model to develop and maintain the TOE.

Content and presentation of evidence elements

ALC_LCD.2.1C The life-cycle definition documentation shall describe the model used to develop and maintain the TOE.

ALC_LCD.2.2C The life-cycle model shall provide for the necessary control over the development and maintenance of the TOE.

ALC_LCD.2.3C The life-cycle definition documentation shall explain why the model was chosen.

ALC_LCD.2.4C The life-cycle definition documentation shall explain how the model is used to develop and maintain the TOE.

ALC_LCD.2.5C The life-cycle definition documentation shall demonstrate compliance with the standardised life-cycle model.

Evaluator action elements

ALC_LCD.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ALC_LCD.3 Measurable life-cycle model

Developer action elements

ALC_LCD.3.1D The developer shall establish a life-cycle model to be used in the development and maintenance of the TOE.

ALC_LCD.3.2D The developer shall provide life-cycle definition documentation.

ALC_LCD.3.3D The developer shall use a standardised and measurable life-cycle model to develop and maintain the TOE.

ALC_LCD.3.4D The developer shall measure the TOE development using the standardised and measurable life-cycle model.

Content and presentation of evidence elements

ALC_LCD.3.1C The life-cycle definition documentation shall describe the model used to develop and maintain the TOE, including the details of its arithmetic parameters and/or metrics used to measure the TOE development against the model.

ALC_LCD.3.2C The life-cycle model shall provide for the necessary control over the development and maintenance of the TOE.

ALC_LCD.3.3C The life-cycle definition documentation shall explain why the model was chosen.

ALC_LCD.3.4C The life-cycle definition documentation shall explain how the model is used to develop and maintain the TOE.

ALC_LCD.3.5C The life-cycle definition documentation shall demonstrate compliance with the standardised and measurable life-cycle model.

ALC_LCD.3.6C The life-cycle documentation shall provide the results of the measurements of the TOE development using the standardised and measurable life-cycle model.

Evaluator action elements

ALC_LCD.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

10.4 Tools and techniques (ALC_TAT)

Objectives

362 Tools and techniques is an aspect of selecting tools that are used to develop, analyse and implement the TOE. It includes requirements to prevent ill-defined, inconsistent or incorrect development tools from being used to develop the TOE. This includes, but is not limited to, programming languages, documentation, implementation standards, and other parts of the TOE such as supporting runtime libraries.

Component levelling

363 The components in this family are levelled on the basis of increasing requirements on the description and scope of the implementation standards and the documentation of implementation- dependent options.

Application notes

364 There is a requirement for well-defined development tools. These are tools that have been shown to be applicable without the need for intensive further clarification. For example, programming languages and computer aided design (CAD) systems that are based on an a standard published by standards bodies are considered to be well-defined.

365 Tools and techniques distinguishes between the implementation standards applied by the developer (ALC_TAT.2.3D) and the implementation standards for "all parts of the TOE" (ALC_TAT.3.3D) that additionally includes third party software, hardware, or firmware.

366 The requirement in ALC_TAT.1.2C is especially applicable to programming languages so as to ensure that all statements in the source code have an unambiguous meaning.

ALC_TAT.1 Well-defined development tools

Dependencies

ADV_IMP.1 Subset of the implementation of the TSF

Developer action elements

ALC_TAT.1.1D The developer shall identify the development tools being used for the TOE.

ALC_TAT.1.2D The developer shall document the selected implementation-dependent options of the development tools.

Content and presentation of evidence elements

ALC_TAT.1.1C All development tools used for implementation shall be well-defined.

ALC_TAT.1.2C The documentation of the development tools shall unambiguously define the meaning of all statements used in the implementation.

ALC_TAT.1.3C The documentation of the development tools shall unambiguously define the meaning of all implementation-dependent options.

Evaluator action elements

ALC_TAT.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ALC_TAT.2 Compliance with implementation standards

Dependencies

ADV_IMP.1 Subset of the implementation of the TSF

Developer action elements

ALC_TAT.2.1D The developer shall identify the development tools being used for the TOE.

ALC_TAT.2.2D The developer shall document the selected implementation-dependent options of the development tools.

ALC_TAT.2.3D The developer shall describe the implementation standards to be applied.

Content and presentation of evidence elements

ALC_TAT.2.1C All development tools used for implementation shall be well-defined.

ALC_TAT.2.2C The documentation of the development tools shall unambiguously define the meaning of all statements used in the implementation.

ALC_TAT.2.3C The documentation of the development tools shall unambiguously define the meaning of all implementation-dependent options.

Evaluator action elements

ALC_TAT.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ALC_TAT.2.2E The evaluator shall confirm that the implementation standards have been applied.

ALC_TAT.3 Compliance with implementation standards - all parts

Dependencies

ADV_IMP.1 Subset of the implementation of the TSF

Developer action elements

ALC_TAT.3.1D The developer shall identify the development tools being used for the TOE.

ALC_TAT.3.2D The developer shall document the selected implementation-dependent options of the development tools.

ALC_TAT.3.3D The developer shall describe the implementation standards for all parts of the TOE.

Content and presentation of evidence elements

ALC_TAT.3.1C All development tools used for implementation shall be well-defined.

ALC_TAT.3.2C The documentation of the development tools shall unambiguously define the meaning of all statements used in the implementation.

ALC_TAT.3.3C The documentation of the development tools shall unambiguously define the meaning of all implementation-dependent options.

Evaluator action elements

ALC_TAT.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ALC_TAT.3.2E The evaluator shall confirm that the implementation standards have been applied.

11 Class ASE: Security Target evaluation

367 Evaluating an ST is required to demonstrate that the ST is sound and internally consistent, and, if the ST is based on one or more PPs or packages, that the ST is a correct instantiation of these PPs and packages. These properties are necessary for the ST to be suitable for use as the basis for the rest of the TOE evaluation.

368 Figure **13** shows the families within this class, and the hierarchy of components within the families.

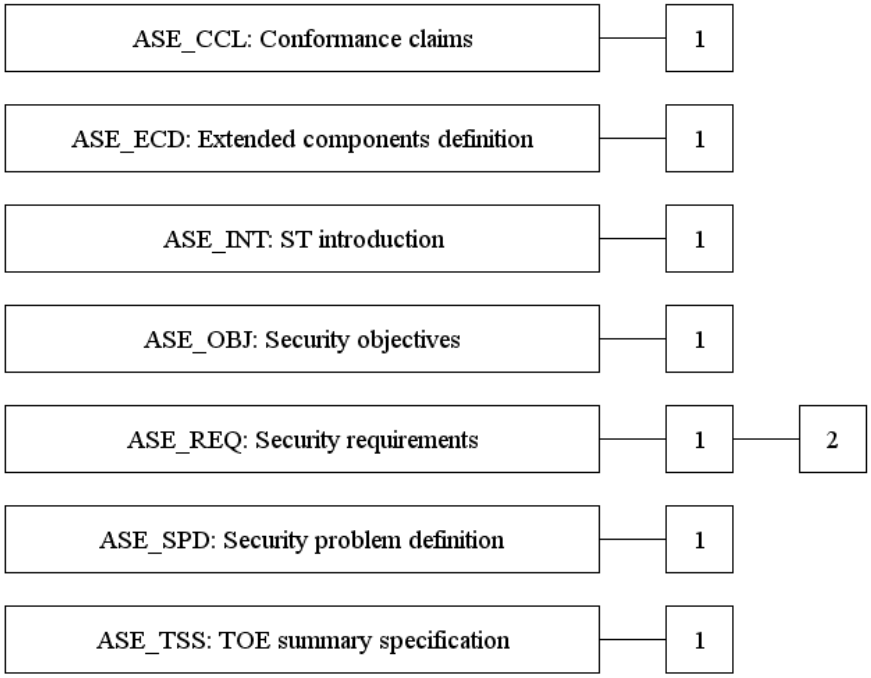

Figure 13 - ASE: Security Target evaluation class decomposition

11.1 Conformance claims (ASE_CCL)

Objectives

369 The objective of this family is to determine the validity of the conformance claim. In addition, this family specifies how STs are to claim conformance with the PP.

ASE_CCL.1 Conformance claims

Dependencies

ASE_INT.1 ST introduction
ASE_SPD.1 Security problem definition
ASE_OBJ.1 Security objectives
ASE_ECD.1 Extended components definition

ASE_REQ.1 Stated security requirements

Developer action elements

ASE_CCL.1.1D The developer shall provide a conformance claim.

ASE_CCL.1.2D The developer shall provide a conformance claim rationale.

Content and presentation of evidence elements

ASE_CCL.1.1C The conformance claim shall contain a CC conformance claim that identifies the version of the CC to which the ST and the TOE claim conformance.

ASE_CCL.1.2C The CC conformance claim shall describe the conformance of the ST to CC Part 2 as either CC Part 2 conformant or CC Part 2 extended.

ASE_CCL.1.3C The CC conformance claim shall describe the conformance of the ST to CC Part 3 as either CC Part 3 conformant or CC Part 3 extended.

ASE_CCL.1.4C The CC conformance claim shall be consistent with the extended components definition.

ASE_CCL.1.5C The conformance claim shall identify all PPs and security requirement packages to which the ST claims conformance.

ASE_CCL.1.6C The conformance claim shall describe any conformance of the ST to a package as either package-conformant or package-augmented.

ASE_CCL.1.7C The conformance claims rationale shall demonstrate that the TOE type is consistent with the TOE type in the PPs for which conformance is being claimed.

ASE_CCL.1.8C The conformance claims rationale shall demonstrate that the statement of the security problem definition is consistent with the statement of the security problem definition in the PPs for which conformance is being claimed.

ASE_CCL.1.9C The conformance claims rationale shall demonstrate that the statement of objectives is consistent with the statement of objectives in the PPs for which conformance is being claimed.

ASE_CCL.1.10C The conformance claims rationale shall demonstrate that the statement of security requirements is consistent with the statement of security requirements in the PPs for which conformance is being claimed.

ASE_CCL.1.11C The conformance claims rationale shall demonstrate that all operations of the security requirements that were taken from a PP are completed consistently with the respective PP.

ASE_CCL.1.12C The conformance claims rationale shall demonstrate that the statement of security requirements is consistent with the statement of security requirements in the security requirement package for which conformance is being claimed.

ASE_CCL.1.13C The conformance claims rationale shall demonstrate that all operations of the security requirements in the ST that were taken from a package are completed consistently with the respective security requirement package.

ASE_CCL.1.14C The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

Evaluator action elements

ASE_CCL.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

11.2 Extended components definition (ASE_ECD)

Objectives

370 Extended security requirements are requirements that are not based on components from CC Part 2 or CC Part 3, but are based on extended components: components defined by the ST author.

371 Evaluation of the definition of extended components is necessary to determine that they are clear and unambiguous, and that they are necessary, i.e. they could not have been clearly expressed using existing CC Part 2 or CC Part 3 components.

ASE_ECD.1 Extended components definition

Developer action elements

ASE_ECD.1.1D The developer shall provide a statement of security requirements.

ASE_ECD.1.2D The developer shall provide an extended components definition.

Content and presentation of evidence elements

ASE_ECD.1.1C The statement of security requirements shall identify all extended security requirements.

ASE_ECD.1.2C The extended components definition shall define an extended component for each extended security requirement.

ASE_ECD.1.3C The extended components definition shall describe how each extended component is related to the existing CC components, families, and classes.

ASE_ECD.1.4C The extended components definition shall use the existing CC components, families, classes, and methodology as a model for presentation.

ASE_ECD.1.5C The extended components shall consist of measurable and objective elements such that compliance or noncompliance to these elements can be demonstrated.

Evaluator action elements

ASE_ECD.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ASE_ECD.1.2E The evaluator shall confirm that no extended component can be clearly expressed using existing components.

11.3 ST introduction (ASE_INT)

Objectives

372 The objective of this family is to describe the TOE in a narrative way on three levels of abstraction: ST/TOE reference, TOE overview and TOE description.

373 Evaluation of the ST introduction is required to demonstrate that the ST and the TOE are correctly identified, that the TOE is correctly described at three levels of abstraction and that these three descriptions are consistent with each other.

ASE_INT.1 ST introduction

Developer action elements

ASE_INT.1.1D The developer shall provide an ST introduction.

Content and presentation of evidence elements

ASE_INT.1.1C The ST introduction shall contain an ST reference, a TOE reference, a TOE overview and a TOE description.

ASE_INT.1.2C The ST reference shall uniquely identify the ST.

ASE_INT.1.3C The TOE reference shall identify the TOE.

ASE_INT.1.4C The TOE overview shall summarise the usage and major security features of the TOE.

ASE_INT.1.5C The TOE overview shall identify the TOE type.

ASE_INT.1.6C The TOE overview shall identify any non-TOE hardware/software/firmware required by the TOE.

ASE_INT.1.7C The TOE description shall describe the physical scope and boundaries of the TOE.

ASE_INT.1.8C The TOE description shall describe the logical scope and boundaries of the TOE.

Evaluator action elements

ASE_INT.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ASE_INT.1.2E The evaluator shall confirm that the TOE reference, the TOE overview, and the TOE description are consistent with each other.

11.4 Security objectives (ASE_OBJ)

Objectives

374 The security objectives are a concise statement of the intended response to the security problem defined through the Security problem definition (ASE_SPD) family.

375 Evaluation of the security objectives is required to demonstrate that the security objectives adequately and completely address the security problem definition, that the division of this problem between the TOE, its development environment, and its operational environment is clearly defined, and that the security objectives are internally consistent.

ASE_OBJ.1 Security objectives

Dependencies

ASE_SPD.1 Security problem definition

Developer action elements

ASE_OBJ.1.1D The developer shall provide a statement of security objectives.

ASE_OBJ.1.2D The developer shall provide a security objectives rationale.

Content and presentation of evidence elements

ASE_OBJ.1.1C The statement of security objectives shall describe the security objectives for the TOE.

ASE_OBJ.1.2C The security objectives rationale shall trace each security objective for the TOE back to threats countered by that security objective and OSPs met by that security objective.

ASE_OBJ.1.3C The statement of security objectives shall describe the security objectives for the development environment.

ASE_OBJ.1.4C The security objectives rationale shall trace each security objective for the development environment back to threats countered by that security objective and OSPs met by that security objective.

ASE_OBJ.1.5C The statement of security objectives shall describe the security objectives for the operational environment

ASE_OBJ.1.6C The security objectives rationale shall trace each security objective for the operational environment back to threats countered by that security objective, OSPs enforced by that security objective, and assumptions upheld by that security objective.

ASE_OBJ.1.7C The security objectives rationale shall demonstrate that the security objectives counter all threats.

ASE_OBJ.1.8C The security objectives rationale shall demonstrate that the security objectives enforce all OSPs.

ASE_OBJ.1.9C The security objectives rationale shall demonstrate that the security objectives for the operational environment uphold all assumptions.

Evaluator action elements

ASE_OBJ.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ASE_OBJ.1.2E The evaluator shall confirm that the statement of security objectives is internally consistent.

11.5 Security requirements (ASE_REQ)

Objectives

376 The SFRs form a clear, unambiguous and canonical description of the expected security behaviour of the TOE. The SARs form a clear, unambiguous and canonical description of the expected activities that will be undertaken to gain assurance in the TOE.

377 Evaluation of the security requirements is required to ensure that they are clear, unambiguous and canonical.

Component levelling

378 The components in this family are levelled on whether they are stated as is, or whether they are derived from security objectives for the TOE and security objectives for the development environment.

ASE_REQ.1 Stated security requirements

Dependencies

ASE_ECD.1 Extended components definition

Content and presentation of evidence elements

ASE_REQ.1.1C The statement of security requirements shall describe the SFRs and the SARs.

ASE_REQ.1.2C The statement of security requirements shall identify all operations on the security requirements.

ASE_REQ.1.3C All assignment and selection operations shall be completed.

ASE_REQ.1.4C All operations shall be performed correctly.

Evaluator action elements

ASE_REQ.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ASE_REQ.1.2E The evaluator shall confirm that the statement of security requirements is internally consistent.

ASE_REQ.2 Derived security requirements

Dependencies

ASE_OBJ.1 Security objectives
ASE_ECD.1 Extended components definition

Developer action elements

ASE_REQ.2.1D The developer shall provide a security requirements rationale.

Content and presentation of evidence elements

ASE_REQ.2.1C The statement of security requirements shall describe the SFRs and the SARs.

ASE_REQ.2.2C The statement of security requirements shall identify all operations on the security requirements.

ASE_REQ.2.3C All assignment and selection operations shall be completed.

ASE_REQ.2.4C All operations shall be performed correctly.

ASE_REQ.2.5C Each dependency of the security requirements shall either be satisfied, or the security requirements rationale shall justify the dependency not being satisfied.

ASE_REQ.2.6C The security requirements rationale shall trace each SFR back to the security objectives for the TOE.

ASE_REQ.2.7C The security requirements rationale shall demonstrate that the SFRs meet all security objectives for the TOE.

ASE_REQ.2.8C The security requirements rationale shall trace each SAR back to the security objectives for the development environment.

ASE_REQ.2.9C The security requirements rationale shall demonstrate that the SARs meet all security objectives for the development environment.

Evaluator action elements

ASE_REQ.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ASE_REQ.2.2E The evaluator shall confirm that the statement of security requirements is internally consistent.

11.6 Security problem definition (ASE_SPD)

Objectives

379 This part of the ST defines the security problem to be addressed by the TOE, the operational environment of the TOE, and the development environment of the TOE.

380 Evaluation of the security problem definition is required to demonstrate that the security problem intended to be addressed by the TOE, its operational environment, and its development environment, is clearly defined.

ASE_SPD.1 Security problem definition

Developer action elements

ASE_APD.1.1D The developer shall provide a security problem definition.

Content and presentation of evidence elements

ASE_SPD.1.1C The security problem definition shall describe the threats.

ASE_SPD.1.2C All threats shall be described in terms of a threat agent, an asset, and an adverse action.

ASE_SPD.1.3C The security problem definition shall describe the OSPs.

ASE_SPD.1.4C The security problem definition shall describe the assumptions about the operational environment of the TOE.

Evaluator action elements

ASE_SPD.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

11.7 TOE summary specification (ASE_TSS)

Objectives

381 The TOE summary specification allows evaluators and potential consumers to understand how the TOE meets its SFRs.

382 Evaluation of the TOE summary specification is necessary to determine whether all SFRs have been adequately addressed, and whether the TOE summary specification is consistent with other narrative descriptions of the TOE.

ASE_TSS.1 TOE summary specification

Dependencies

ASE_INT.1 ST introduction

ASE_REQ.1 Stated security requirements

Developer action elements

ASE_TSS.1.1D The developer shall provide a TOE summary specification.

Content and presentation of evidence elements

ASE_TSS.1.1C The TOE summary specification shall describe how the TOE meets each SFR.

Evaluator action elements

ASE_TSS.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ASE_TSS.1.2E The evaluator shall confirm that the TOE summary specification is consistent with the TOE overview and the TOE description.

12 Class ATE: Tests

383 The class “Tests” encompasses four families: coverage (ATE_COV), depth (ATE_DPT), independent testing (e.g. functional testing performed by evaluators) (ATE_IND), and functional tests (ATE_FUN). Testing provide assurance that the TSF meets its design descriptions (functional specifications, high-level design, low-level design and implementation representation).

384 The emphasis in this class is on confirmation that the TSF operates according to its design descriptions. This class does not address penetration testing, which is based upon an analysis of the TSF that specifically seeks to identify vulnerabilities in the design and implementation of the TSF. Penetration testing is addressed separately as an aspect of vulnerability assessment in the AVA class.

385 The ATE: Tests class separates testing into developer testing and evaluator testing. The Coverage (ATE_COV) and Depth (ATE_DPT) families address the completeness of developer testing. Coverage (ATE_COV) addresses the rigor with which the functional specification is tested, Depth (ATE_DPT) addresses whether testing against other design descriptions (high-level design, low-level design, implementation representation) is required.

386 Functional tests (ATE_FUN) addresses the performing of these tests by the developer and how this testing should be documented. Finally, Independent testing (ATE_IND) then addresses evaluator testing: whether the evaluator should redo part or all of the developer testing and how much independent testing the evaluator should do.

387 Figure **14** shows the families within this class, and the hierarchy of components within the families.

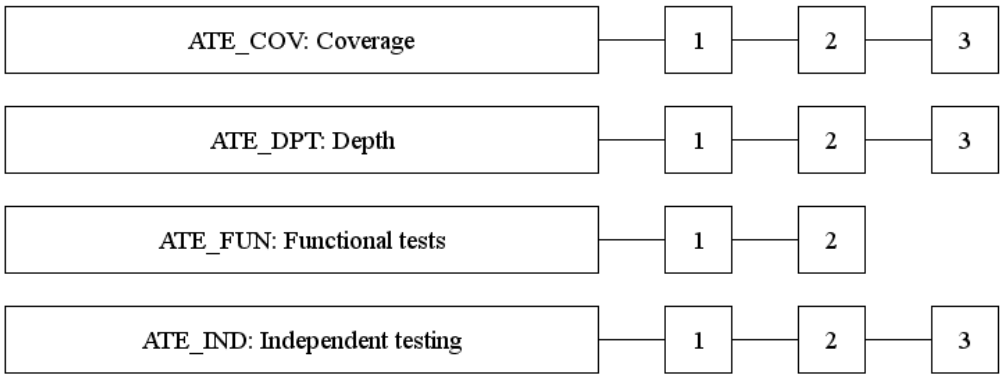

Figure 14 - ATE: Tests class decomposition

12.1 Coverage (ATE_COV)

Objectives

388 This family addresses those aspects of testing that deal with completeness of test coverage. That is, it addresses the extent to which the TSF is tested, and whether or not the testing is sufficiently extensive to demonstrate that the TSF operates in accordance with its functional specification.

Component levelling

389 The components in this family are levelled on the basis of increasing rigour of interface testing, and increasing rigour of the analysis of the sufficiency of the tests to demonstrate that the TSF operates in accordance with its functional specification.

Application notes

390 Not all tests in the test documentation have to correspond to interfaces in the functional specification: some tests may address other interfaces such as internal interfaces that are only visible in the high-level design.

ATE_COV.1 Evidence of coverage

Dependencies

ADV_FSP.1 Informal functional specification
ATE_FUN.1 Functional testing

Objectives

391 In this component, the objective is to confirm that the developer performed some tests of some interfaces in the functional specification. This is to be achieved through an examination of developer evidence of correspondence.

Application notes

392 In this component the developer is required to show how tests in the test documentation correspond to interfaces in the functional specification. This can be achieved by a statement of correspondence, perhaps using a table.

Developer action elements

ATE_COV.1.1D The developer shall provide evidence of the test coverage.

Content and presentation of evidence elements

ATE_COV.1.1C The evidence of the test coverage shall show the correspondence between the tests in the test documentation and the interfaces in the functional specification.

Evaluator action elements

ATE_COV.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ATE_COV.2 Analysis of coverage

Dependencies

ADV_FSP.1 Informal functional specification
ATE_FUN.1 Functional testing

Objectives

393 In this component, the objective is to confirm that the developer performed some tests of all interfaces in the functional specification. This is to be achieved through an examination of developer evidence of correspondence.

Application notes

394 In this component the developer is required to show how tests in the test documentation correspond to interfaces in the functional specification. This can be achieved by a statement of correspondence, perhaps using a table.

Developer action elements

ATE_COV.2.1D The developer shall provide an analysis of the test coverage.

Content and presentation of evidence elements

ATE_COV.2.1C The analysis of the test coverage shall demonstrate the correspondence between the tests in the test documentation and the interfaces in the functional specification.

ATE_COV.2.2C The analysis of the test coverage shall demonstrate that the correspondence between the interfaces in the functional specification and the tests in the test documentation is complete.

Evaluator action elements

ATE_COV.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ATE_COV.3 Rigorous analysis of coverage

Dependencies

ADV_FSP.1 Informal functional specification

ATE_FUN.1 Functional testing

Objectives

395 In this component, the objective is to confirm that the developer performed exhaustive tests of all interfaces in the functional specification.

Application notes

396 In this component the developer is required to show how tests in the test documentation correspond to interfaces in the functional specification. This can be achieved by a statement of correspondence, perhaps using a table, but in addition the developer is required to demonstrate that the tests exhaustively test each interface in the functional specification.

Developer action elements

ATE_COV.3.1D The developer shall provide an analysis of the test coverage.

Content and presentation of evidence elements

ATE_COV.3.1C The analysis of the test coverage shall demonstrate the correspondence between the tests in the test documentation and the interfaces in the functional specification.

ATE_COV.3.2C The analysis of the test coverage shall demonstrate that the correspondence between the interfaces in the functional specification and the tests in the test documentation is complete.

ATE_COV.3.3C The analysis of the test coverage shall rigorously demonstrate that all interfaces in the functional specification have been exhaustively tested.

Evaluator action elements

ATE_COV.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

12.2 Depth (ATE_DPT)

Objectives

397 The components in this family deal with the level of detail to which the TSF is tested. Testing of the TSF is based upon increasing depth of information derived from additional design representations (high-level design, low-level design, implementation representation).

398 The objective is to counter the risk of missing an error in the development of the TOE. Additionally, the components of this family, especially as testing is more concerned with the internal structure of the TSF, are more likely to discover any malicious code that has been inserted.

399 Testing that exercises specific internal interfaces can provide assurance not only that the TSF exhibits the desired external security behaviour, but also that this behaviour stems from correctly operating internal mechanisms.

Component levelling

400 The components in this family are levelled on the basis of increasing detail provided in the TSF representations, from the high-level design to the implementation representation. This levelling reflects the TSF representations presented in the ADV class.

Application notes

401 The high-level design should describe each of the subsystems and also describe the interfaces between these subsystems in sufficient detail. Evidence of testing of this high-level design must show that the internal interfaces between subsystems have been exercised. This may be achieved through testing via the external interfaces of the TSF, or by testing of the subsystem interfaces in isolation, perhaps employing a test harness. In cases where some aspects of an internal interface cannot be tested via the external interfaces there should either be justification that these aspects need not be tested, or the internal interface needs to be tested directly. In the latter case the high-level design needs to be sufficiently detailed in order to facilitate direct testing.

402 A smilar line of reasoning applies to the higher components in this family: they aim to check the correct operation of internal interfaces that become visible as the design becomes less abstract. When these components are applied it will be more difficult to provide adequate evidence of the depth of testing using the TSF's external interfaces alone, and modular testing will usually be necessary.

ATE_DPT.1 Testing: high-level design

Dependencies

ADV_HLD.1 Descriptive high-level design
ATE_FUN.1 Functional testing

Objectives

403 The subsystems of the TSF provide a high-level description of the internal workings of the TSF. Testing at the level of the subsystems provides assurance that the TSF subsystems have been correctly realised.

Developer action elements

ATE_DPT.1.1D The developer shall provide the analysis of the depth of testing.

Content and presentation of evidence elements

ATE_DPT.1.1C The analysis of the depth of testing shall demonstrate the correspondence between the tests in the test documentation and the interfaces in the high-level design.

ATE_DPT.1.2C The analysis of the depth of testing shall demonstrate that the correspondence between the interfaces in the high-level design and the tests in the test documentation is complete.

Evaluator action elements

ATE_DPT.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ATE_DPT.2 Testing: low-level design

Dependencies

ADV_HLD.2 Security enforcing high-level design
ADV_LLD.1 Descriptive low-level design
ATE_FUN.1 Functional testing

Objectives

404 The subsystems of the TSF provide a high-level description of the internal workings of the TSF. Testing at the level of the subsystems provides assurance that the TSF subsystems have been correctly realised.

405 The modules of the TSF provide a description of the internal workings of the TSF. Testing at the level of the modules provides assurance that the TSF modules have been correctly realised.

Developer action elements

ATE_DPT.2.1D The developer shall provide the analysis of the depth of testing.

Content and presentation of evidence elements

ATE_DPT.2.1C The analysis of the depth of testing shall demonstrate the correspondence between the tests in the test documentation and the interfaces in the high-level design and the low-level design.

ATE_DPT.2.2C The analysis of the depth of testing shall demonstrate that the correspondence between the interfaces in the high-level design and the tests in the test documentation is complete.

ATE_DPT.2.3C The analysis of the depth of testing shall demonstrate that the correspondence between the interfaces in the low-level design and the tests in the test documentation is complete.

Evaluator action elements

ATE_DPT.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ATE_DPT.3 Testing: implementation representation

Dependencies

ADV_HLD.2 Security enforcing high-level design
ADV_IMP.2 Implementation of the TSF
ADV_LLD.1 Descriptive low-level design
ATE_FUN.1 Functional testing

Objectives

406 The subsystems of the TSF provide a high-level description of the internal workings of the TSF. Testing at the level of the subsystems provides assurance that the TSF subsystems have been correctly realised.

407 The modules of the TSF provide a description of the internal workings of the TSF. Testing at the level of the modules provides assurance that the TSF modules have been correctly realised.

408 The implementation representation of the TSF provides a detailed description of the internal workings of the TSF. Testing at the level of the implementation provides assurance that the TSF implementation has been correctly realised.

Developer action elements

ATE_DPT.3.1D The developer shall provide the analysis of the depth of testing.

Content and presentation of evidence elements

ATE_DPT.3.1C The analysis of the depth of testing shall demonstrate the correspondence between the tests in the test documentation and the interfaces in the high-level design, the low-level design and the implementation representation.

ATE_DPT.3.2C The analysis of the depth of testing shall demonstrate that the correspondence between the interfaces in the high-level design and the tests in the test documentation is complete.

ATE_DPT.3.3C The analysis of the depth of testing shall demonstrate that the correspondence between the interfaces in the low-level design and the tests in the test documentation is complete.

ATE_DPT.3.4C The analysis of the depth of testing shall demonstrate that the correspondence between the interfaces in the implementation representation and the tests in the test documentation is complete.

Evaluator action elements

ATE_DPT.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

12.3 Functional tests (ATE_FUN)

Objectives

409 Functional testing performed by the developer provides assurance that the tests in the test documentation are performed and documented correctly. The correspondence of these tests to the design descriptions of the TSF is achieved through the Coverage (ATE_COV) and Depth (ATE_DPT) families.

410 This family contributes to providing assurance that the likelihood of undiscovered flaws is relatively small.

411 The families Coverage (ATE_COV), Depth (ATE_DPT) and Functional tests (ATE_FUN) are used in combination to define the evidence of testing to be supplied by a developer. Independent functional testing by the evaluator is specified by Independent testing (ATE_IND).

Component levelling

412 This family contains two components, the higher requiring that ordering dependencies are analysed.

Application notes

413 Procedures for performing tests are expected to provide instructions for using test programs and test suites, including the test environment, test conditions, test data parameters and values. The test procedures should also show how the test results are derived from the test inputs.

414 Ordering dependencies are relevant when the successful execution of a particular test depends upon the existence of a particular state. For example, this might require that test A be executed immediately before test B, since the state resulting from the successful execution of test A is a prerequisite for the successful execution of test B. Thus, failure of test B could be related to a problem with the ordering dependencies. In the above example, test B could fail because test C (rather than test A) was executed immediately before it, or the failure of test B could be related to a failure of test A.

ATE_FUN.1 Functional testing

Objectives

415 The objective is for the developer to demonstrate that the tests in the test documentation are performed and documented correctly.

Developer action elements

ATE_FUN.1.1D The developer shall test the TSF and document the results.

ATE_FUN.1.2D The developer shall provide test documentation.

Content and presentation of evidence elements

ATE_FUN.1.1C The test documentation shall consist of test plans, expected test results and actual test results.

ATE_FUN.1.2C The test plans shall identify the tests to be performed and describe the scenarios for performing each test. These scenarios shall include any ordering dependencies on the results of other tests.

ATE_FUN.1.3C The expected test results shall show the anticipated outputs from a successful execution of the tests.

ATE_FUN.1.4C The actual test results shall be consistent with the expected test results.

Evaluator action elements

ATE_FUN.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ATE_FUN.2 Ordered functional testing

Objectives

416 The objectives are for the developer to demonstrate that the tests in the test documentation are performed and documented correctly, and to ensure that testing is structured such as to avoid circular arguments about the correctness of the interfaces being tested.

Application notes

417 Although the test procedures may state pre-requisite initial test conditions in terms of ordering of tests, they may not provide a rationale for the ordering. An analysis of test ordering is an important factor in determining the adequacy of testing, as there is a possibility of faults being concealed by the ordering of tests.

Developer action elements

ATE_FUN.2.1D The developer shall test the TSF and document the results.

ATE_FUN.2.2D The developer shall provide test documentation.

Content and presentation of evidence elements

ATE_FUN.2.1C The test documentation shall consist of test plans, expected test results and actual test results.

ATE_FUN.2.2C The test plans shall identify the tests to be performed and describe the scenarios for performing each test. These scenarios shall include any ordering dependencies on the results of other tests.

ATE_FUN.2.3C The expected test results shall show the anticipated outputs from a successful execution of the tests.

ATE_FUN.2.4C The actual test results shall be consistent with the expected test results.

ATE_FUN.2.5C The test documentation shall include an analysis of the test procedure ordering dependencies.

Evaluator action elements

ATE_FUN.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

12.4 Independent testing (ATE_IND)

Objectives

418 The objectives are to gain more assurance in that the TSF meets its design representations by verifying the developer testing and the performing of additional tests by the evaluator.

Component levelling

419 Levelling is based upon the amount of test documentation, test support and the amount of evaluator testing.

Application notes

420 This family deals with the degree to which there is independent functional testing of the TSF. Independent functional testing may take the form of repeating the developer's functional tests, in whole or in part. It may also take the form of the augmentation of the developer's functional tests, either to extend the scope or the depth of the developer's tests, or to test for obvious public domain security weaknesses that could be applicable to the TOE. These activities are complementary, and an appropriate mix must be planned for each TOE, which takes into account the availability and coverage of test results, and the functional complexity of the TSF.

421 Sampling of developer tests is intended to provide confirmation that the developer has carried out his planned test programme on the TSF, and has correctly recorded the results. The size of sample selected will be influenced

by the detail and quality of the developer's functional test results. The evaluator will also need to consider the scope for devising additional tests, and the relative benefit that may be gained from effort in these two areas. It is recognised that repetition of all developer tests may be feasible and desirable in some cases, but may be very arduous and less productive in others. The highest component in this family should therefore be used with caution. Sampling will address the whole range of test results available, including those supplied to meet the requirements of both Coverage (ATE_COV) and Depth (ATE_DPT).

422 There is also a need to consider the different configurations of the TOE that are included within the evaluation. The evaluator will need to assess the applicability of the results provided, and to plan his own testing accordingly.

423 The suitability of the TOE for testing is based on the access to the TOE, and the supporting documentation and information required (including any test software or tools) to run tests. The need for such support is addressed by the dependencies to other assurance families.

424 Additionally, suitability of the TOE for testing may be based on other considerations. For example, the version of the TOE submitted by the developer may not be the final version.

425 The term interfaces refers to interfaces in the functional specification, high-level design, low-level design or implementation representation. The exact set of interfaces to be used is selected through the Depth (ATE_DPT) component.

426 References to a subset of the interfaces are intended to allow the evaluator to design an appropriate set of tests which is consistent with the objectives of the evaluation being conducted.

ATE_IND.1 Independent testing - conformance

Dependencies

ADV_FSP.1 Informal functional specification
AGD_ADM.1 Administrator guidance
AGD_USR.1 User guidance

Objectives

427 In this component, the objective is to demonstrate that the TOE operates at least partially in accordance with its design representations.

Application notes

428 This component does not address the use of developer test results. It is applicable where such results are not available, and also in cases where the developer's testing is accepted without validation. The evaluator is required to devise and conduct tests with the objective of confirming that the TOE

operates in accordance with its design representations. The approach is to gain confidence in correct operation through representative testing, rather than to conduct every possible test. The extent of testing to be planned for this purpose is a methodology issue, and needs to be considered in the context of a particular TOE and the balance of other evaluation activities.

Developer action elements

ATE_IND.1.1D The developer shall provide the TOE for testing.

Content and presentation of evidence elements

ATE_IND.1.1C The TOE shall be suitable for testing.

Evaluator action elements

ATE_IND.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ATE_IND.1.2E The evaluator shall test the TSF to confirm that the TSF operates as specified.

ATE_IND.2 Independent testing - sample

Dependencies

ADV_FSP.1 Informal functional specification
AGD_ADM.1 Administrator guidance
AGD_USR.1 User guidance
ATE_FUN.1 Functional testing

Objectives

429 In this component, the objective is to confirm that the developer performed some tests of some interfaces in the functional specification. This is to be achieved through an examination of developer evidence of correspondence.

Application notes

430 The intent is that the developer should provide the evaluator with materials necessary for the efficient reproduction of developer tests. This may include such things as machine-readable test documentation, test programs, etc.

431 This component contains a requirement that the evaluator has available test results from the developer to supplement the programme of testing. The evaluator will repeat a sample of the developer's tests to gain confidence in the results obtained. Having established such confidence the evaluator will build upon the developer's testing by conducting additional tests that exercise the TOE in a different manner. By using a platform of validated developer

test results the evaluator is able to gain confidence that the TOE operates correctly in a wider range of conditions than would be possible purely using the developer's own efforts, given a fixed level of resource. Having gained confidence that the developer has tested the TOE, the evaluator will also have more freedom, where appropriate, to concentrate testing in areas where examination of documentation or specialist knowledge has raised particular concerns.

Developer action elements

ATE_IND.2.1D The developer shall provide the TOE for testing.

Content and presentation of evidence elements

ATE_IND.2.1C The TOE shall be suitable for testing.

ATE_IND.2.2C The developer shall provide an equivalent set of resources to those that were used in the developer's functional testing of the TSF.

Evaluator action elements

ATE_IND.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ATE_IND.2.2E The evaluator shall execute a sample of tests in the test documentation to verify the developer test results.

ATE_IND.2.3E The evaluator shall test the TSF to confirm that the TSF operates as specified.

ATE_IND.3 Independent testing - complete

Dependencies

ADV_FSP.1 Informal functional specification
AGD_ADM.1 Administrator guidance
AGD_USR.1 User guidance
ATE_FUN.1 Functional testing

Objectives

432 In this component, the objective is to demonstrate that the TOE operates in accordance with its design representations. Evaluator testing includes repeating all of the developer tests.

Application notes

433 The intent is that the developer should provide the evaluator with materials necessary for the efficient reproduction of developer tests. This may include such things as machine-readable test documentation, test programs, etc.

434 In this component the evaluator must repeat all of the developer's tests as part of the programme of testing. As in the previous component the evaluator will also conduct tests that aim to exercise the TSF in a different manner from that achieved by the developer. In cases where developer testing has been exhaustive, there may remain little scope for this.

Developer action elements

ATE_IND.3.1D The developer shall provide the TOE for testing.

Content and presentation of evidence elements

ATE_IND.3.1C The TOE shall be suitable for testing.

ATE_IND.3.2C The developer shall provide an equivalent set of resources to those that were used in the developer's functional testing of the TSF.

Evaluator action elements

ATE_IND.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ATE_IND.3.2E The evaluator shall execute all tests in the test documentation to verify the developer test results.

ATE_IND.3.3E The evaluator shall test the TSF to confirm that the TSF operates as specified.

13 Class AVA: Vulnerability assessment

435 The class addresses the existence of exploitable covert channels, the possibility of misuse or incorrect configuration of the TOE and the possibility of exploitable vulnerabilities introduced in the development or the operation of the TOE.

436 Figure **15** shows the families within this class, and the hierarchy of components within the families.

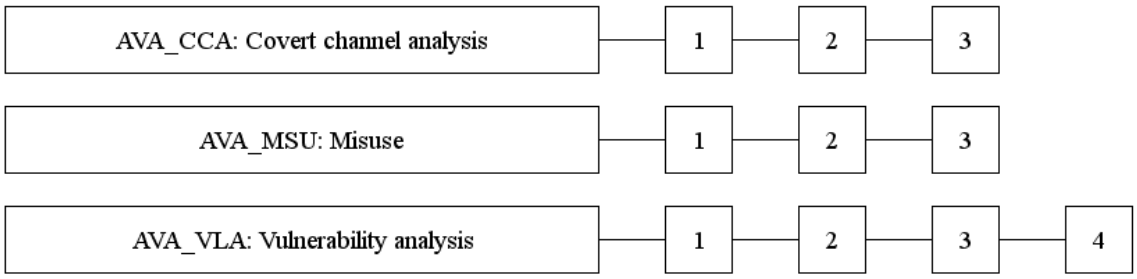

Figure 15 - AVA: Vulnerability assessment class decomposition

13.1 Covert channel analysis (AVA_CCA)

Objectives

437 Covert channel analysis is carried out to determine the existence and potential capacity of unintended signalling channels (i.e. illicit information flows) that may be exploited.

438 The assurance requirements address the threat that unintended and exploitable signalling paths exist that may be exercised to violate the SFP.

Component levelling

439 The components are levelled on increasing rigour of covert channel analysis.

AVA_CCA.1 Covert channel analysis

Dependencies

ADV_FSP.2 Fully defined external interfaces
ADV_IMP.2 Implementation of the TSF
AGD_ADM.1 Administrator guidance
AGD_USR.1 User guidance

Objectives

440 The objective is to identify covert channels that are identifiable, through an informal search for covert channels.

Developer action elements

AVA_CCA.1.1D The developer shall conduct a search for covert channels for each information flow control policy.

AVA_CCA.1.2D The developer shall provide covert channel analysis documentation.

Content and presentation of evidence elements

AVA_CCA.1.1C The analysis documentation shall identify covert channels and estimate their capacity.

AVA_CCA.1.2C The analysis documentation shall describe the procedures used for determining the existence of covert channels, and the information needed to carry out the covert channel analysis.

AVA_CCA.1.3C The analysis documentation shall describe all assumptions made during the covert channel analysis.

AVA_CCA.1.4C The analysis documentation shall describe the method used for estimating channel capacity, based on worst case scenarios.

AVA_CCA.1.5C The analysis documentation shall describe the worst case exploitation scenario for each identified covert channel.

Evaluator action elements

AVA_CCA.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_CCA.1.2E The evaluator shall confirm that the results of the covert channel analysis show that the TOE meets its functional requirements.

AVA_CCA.1.3E The evaluator shall selectively validate the covert channel analysis through testing.

AVA_CCA.2 Systematic covert channel analysis

Dependencies

ADV_FSP.2 Fully defined external interfaces
ADV_IMP.2 Implementation of the TSF
AGD_ADM.1 Administrator guidance
AGD_USR.1 User guidance

Objectives

441 The objective is to identify covert channels that are identifiable, through a systematic search for covert channels.

Developer action elements

AVA_CCA.2.1D The developer shall conduct a search for covert channels for each information flow control policy.

AVA_CCA.2.2D The developer shall provide covert channel analysis documentation.

Content and presentation of evidence elements

AVA_CCA.2.1C The analysis documentation shall identify covert channels and estimate their capacity.

AVA_CCA.2.2C The analysis documentation shall describe the procedures used for determining the existence of covert channels, and the information needed to carry out the covert channel analysis.

AVA_CCA.2.3C The analysis documentation shall describe all assumptions made during the covert channel analysis.

AVA_CCA.2.4C The analysis documentation shall describe the method used for estimating channel capacity, based on worst case scenarios.

AVA_CCA.2.5C The analysis documentation shall describe the worst case exploitation scenario for each identified covert channel.

AVA_CCA.2.6C The analysis documentation shall provide evidence that the method used to identify covert channels is systematic.

AVA_CCA.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_CCA.2.2E The evaluator shall confirm that the results of the covert channel analysis show that the TOE meets its functional requirements.

AVA_CCA.2.3E The evaluator shall selectively validate the covert channel analysis through testing.

AVA_CCA.3 Exhaustive covert channel analysis

Dependencies

ADV_FSP.2 Fully defined external interfaces
ADV_IMP.2 Implementation of the TSF
AGD_ADM.1 Administrator guidance
AGD_USR.1 User guidance

Objectives

442 The objective is to identify covert channels that are identifiable, through an exhaustive search for covert channels.

Developer action elements

AVA_CCA.3.1D The developer shall conduct a search for covert channels for each information flow control policy.

AVA_CCA.3.2D The developer shall provide covert channel analysis documentation.

Content and presentation of evidence elements

AVA_CCA.3.1C The analysis documentation shall identify covert channels and estimate their capacity.

AVA_CCA.3.2C The analysis documentation shall describe the procedures used for determining the existence of covert channels, and the information needed to carry out the covert channel analysis.

AVA_CCA.3.3C The analysis documentation shall describe all assumptions made during the covert channel analysis.

AVA_CCA.3.4C The analysis documentation shall describe the method used for estimating channel capacity, based on worst case scenarios.

AVA_CCA.3.5C The analysis documentation shall describe the worst case exploitation scenario for each identified covert channel.

AVA_CCA.3.6C The analysis documentation shall provide evidence that the method used to identify covert channels is exhaustive.

Evaluator action elements

AVA_CCA.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_CCA.3.2E The evaluator shall confirm that the results of the covert channel analysis show that the TOE meets its functional requirements.

AVA_CCA.3.3E The evaluator shall selectively validate the covert channel analysis through testing.

13.2 Misuse (AVA_MSU)

Objectives

443 Misuse investigates whether the TOE can be configured or used in a manner that is insecure but that an administrator or user of the TOE would reasonably believe to be secure.

444 The objectives are:

a) to minimise the probability of configuring or installing the TOE in a way that is insecure, without the user or administrator being able to detect it;

b) to minimise the risk of human or other errors in operation that may deactivate, disable, or fail to activate the TSF, resulting in an undetected insecure state.

Component levelling

445 The components are levelled on the increasing evidence to be provided by the developer and the increasing rigour of analysis.

AVA_MSU.1 Examination of guidance

Dependencies

ADO_IGS.1 Installation, generation, and start-up procedures
ADV_FSP.1 Informal functional specification
AGD_ADM.1 Administrator guidance
AGD_USR.1 User guidance

Objectives

446 The objective is to ensure that misleading, unreasonable and conflicting guidance is absent from the guidance documentation, and that secure procedures for all modes of operation have been addressed. Insecure states should be easy to detect.

Developer action elements

AVA_MSU.1.1D The developer shall provide guidance documentation.

Content and presentation of evidence elements

AVA_MSU.1.1C The guidance documentation shall identify all possible modes of operation of the TOE (including operation following failure or operational error), their consequences and implications for maintaining secure operation.

AVA_MSU.1.2C The guidance documentation shall be complete, clear and reasonable.

AVA_MSU.1.3C The guidance documentation shall list all security objectives for the operational environment.

Evaluator action elements

AVA_MSU.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_MSU.1.2E The evaluator shall repeat all configuration and installation procedures to confirm that the TOE can be configured and used securely using only the supplied guidance documentation.

AVA_MSU.1.3E The evaluator shall determine that the use of the guidance documentation allows all insecure states to be detected.

AVA_MSU.2 Validation of analysis

Dependencies

ADO_IGS.1 Installation, generation, and start-up procedures
ADV_FSP.1 Informal functional specification
AGD_ADM.1 Administrator guidance
AGD_USR.1 User guidance

Objectives

447 The objective is to ensure that misleading, unreasonable and conflicting guidance is absent from the guidance documentation, and that secure procedures for all modes of operation have been addressed. Insecure states should be easy to detect. In this component, an analysis of the guidance documentation by the developer is required to provide additional assurance that the objective has been met.

Developer action elements

AVA_MSU.2.1D The developer shall provide guidance documentation.

AVA_MSU.2.2D The developer shall document an analysis of the guidance documentation.

Content and presentation of evidence elements

AVA_MSU.2.1C The guidance documentation shall identify all possible modes of operation of the TOE (including operation following failure or operational error), their consequences and implications for maintaining secure operation.

AVA_MSU.2.2C The guidance documentation shall be complete, clear and reasonable.

AVA_MSU.2.3C The guidance documentation shall list all security objectives for the operational environment.

AVA_MSU.2.4C The analysis documentation shall demonstrate that the guidance documentation is complete.

Evaluator action elements

AVA_MSU.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_MSU.2.2E The evaluator shall repeat all configuration and installation procedures, and other procedures selectively, to confirm that the TOE can be configured and used securely using only the supplied guidance documentation.

AVA_MSU.2.3E The evaluator shall determine that the use of the guidance documentation allows all insecure states to be detected.

AVA_MSU.2.4E The evaluator shall confirm that the analysis documentation shows that guidance is provided for secure operation in all modes of operation of the TOE.

AVA_MSU.3 Analysis and testing for insecure states

Dependencies

ADO_IGS.1 Installation, generation, and start-up procedures
ADV_FSP.1 Informal functional specification
AGD_ADM.1 Administrator guidance
AGD_USR.1 User guidance

Objectives

448 The objective is to ensure that misleading, unreasonable and conflicting guidance is absent from the guidance documentation, and that secure procedures for all modes of operation have been addressed. Insecure states should be easy to detect. In this component, an analysis of the guidance documentation by the developer is required to provide additional assurance that the objective has been met, and this analysis is validated and confirmed through testing by the evaluator.

Developer action elements

AVA_MSU.3.1D The developer shall provide guidance documentation.

AVA_MSU.3.2D The developer shall document an analysis of the guidance documentation.

Content and presentation of evidence elements

AVA_MSU.3.1C The guidance documentation shall identify all possible modes of operation of the TOE (including operation following failure or operational error), their consequences and implications for maintaining secure operation.

AVA_MSU.3.2C The guidance documentation shall be complete, clear and reasonable.

AVA_MSU.3.3C The guidance documentation shall list all security objectives for the operational environment.

AVA_MSU.3.4C The analysis documentation shall demonstrate that the guidance documentation is complete.

Evaluator action elements

AVA_MSU.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_MSU.3.2E The evaluator shall repeat all configuration and installation procedures, and other procedures selectively, to confirm that the TOE can be configured and used securely using only the supplied guidance documentation.

AVA_MSU.3.3E The evaluator shall determine that the use of the guidance documentation allows all insecure states to be detected.

AVA_MSU.3.4E The evaluator shall confirm that the analysis documentation shows that guidance is provided for secure operation in all modes of operation of the TOE.

AVA_MSU.3.5E The evaluator ***shall perform*** independent testing to determine that an administrator or user, with an understanding of the guidance documentation, would reasonably be able to determine if the TOE is configured and operating in a manner that is insecure.

13.3 Vulnerability analysis (AVA_VLA)

Objectives

449 Vulnerability analysis is an assessment to determine whether potential vulnerabilities identified, during the evaluation of the construction and anticipated operation of the TOE or by other methods (e.g. by flaw hypotheses), could allow users to violate the TSP.

450 Vulnerability analysis deals with the threats that a user will be able to discover flaws that will allow unauthorised access to resources (e.g. data), allow the ability to interfere with or alter the TSF, or interfere with the authorised capabilities of other users.

Component levelling

451 Levelling is based on an increasing rigour of vulnerability analysis by the developer and the evaluator.

AVA_VLA.1 Developer vulnerability analysis

Objectives

452 A vulnerability analysis is performed by the developer to ascertain the presence of potential vulnerabilities, and to confirm that they cannot be exploited by an attacker with basic attack potential in the operational environment for the TOE.

Developer action elements

AVA_VLA.1.1D The developer shall perform a vulnerability analysis.

AVA_VLA.1.2D The developer shall provide vulnerability analysis documentation.

Content and presentation of evidence elements

AVA_VLA.1.1C The vulnerability analysis documentation shall describe the analysis of the TOE deliverables performed to search for ways in which a user can violate the TSP.

AVA_VLA.1.2C The vulnerability analysis documentation shall describe the disposition of identified potential vulnerabilities.

AVA_VLA.1.3C The vulnerability analysis documentation shall show, for all identified potential vulnerabilities, that the vulnerability cannot be exploited in the operational environment for the TOE.

Evaluator action elements

AVA_VLA.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_VLA.1.2E The evaluator ***shall conduct*** penetration testing, verifying the developer vulnerability analysis and considering encountered potential vulnerabilities, to determine that the TOE in its operational environment is resistant to penetration attacks performed by an attacker possessing basic attack potential.

AVA_VLA.2 Independent vulnerability analysis

Objectives

453 A vulnerability analysis is performed by the developer to ascertain the presence of potential vulnerabilities, and to confirm that they cannot be exploited by an attacker with basic attack potential in the operational environment for the TOE.

454 The evaluator performs independent penetration testing, supported by the evaluator's independent vulnerability analysis, to determine that the TOE is resistant to penetration attacks performed by attackers possessing a basic attack potential.

Developer action elements

AVA_VLA.2.1D The developer shall perform a vulnerability analysis.

AVA_VLA.2.2D The developer shall provide vulnerability analysis documentation.

Content and presentation of evidence elements

AVA_VLA.2.1C The vulnerability analysis documentation shall describe the analysis of the TOE deliverables performed to search for ways in which a user can violate the TSP.

AVA_VLA.2.2C The vulnerability analysis documentation shall describe the disposition of identified potential vulnerabilities.

AVA_VLA.2.3C The vulnerability analysis documentation shall show, for all identified potential vulnerabilities, that the vulnerability cannot be exploited in the operational environment for the TOE.

Evaluator action elements

AVA_VLA.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_VLA.2.2E The evaluator ***shall conduct*** penetration testing, using both the developer vulnerability analysis and the evaluator's independent vulnerability analysis, to determine that the TOE in its operational environment is resistant to penetration attacks performed by an attacker possessing basic attack potential.

AVA_VLA.3 Moderately resistant

Objectives

455 A vulnerability analysis is performed by the developer to ascertain the presence of potential vulnerabilities, and to confirm that they cannot be exploited by an attacker with moderate attack potential in the operational environment for the TOE.

456 The evaluator performs independent penetration testing, supported by the evaluator's independent vulnerability analysis, to determine that the TOE is resistant to penetration attacks performed by attackers possessing a moderate attack potential.

Developer action elements

AVA_VLA.3.1D The developer shall perform a vulnerability analysis.

AVA_VLA.3.2D The developer shall provide vulnerability analysis documentation.

Content and presentation of evidence elements

AVA_VLA.3.1C The vulnerability analysis documentation shall describe the analysis of the TOE deliverables performed to search for ways in which a user can violate the TSP.

AVA_VLA.3.2C The vulnerability analysis documentation shall describe the disposition of identified potential vulnerabilities.

AVA_VLA.3.3C The vulnerability analysis documentation shall show, for all identified potential vulnerabilities, that the vulnerability cannot be exploited in the operational environment for the TOE.

AVA_VLA.3.4C The vulnerability analysis documentation shall show that the search for potential vulnerabilities is systematic.

Evaluator action elements

AVA_VLA.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_VLA.3.2E The evaluator ***shall conduct*** penetration testing, using both the developer vulnerability analysis and the evaluator's independent vulnerability analysis, to determine that the TOE in its operational environment is resistant to penetration attacks performed by an attacker possessing moderate attack potential.

AVA_VLA.4 Highly resistant

Objectives

457 A vulnerability analysis is performed by the developer to ascertain the presence of potential vulnerabilities, and to confirm that they cannot be exploited by an attacker with high attack potential in the operational environment for the TOE.

Developer action elements

AVA_VLA.4.1D The developer shall perform a vulnerability analysis.

AVA_VLA.4.2D The developer shall provide vulnerability analysis documentation.

Content and presentation of evidence elements

AVA_VLA.4.1C The vulnerability analysis documentation shall describe the analysis of the TOE deliverables performed to search for ways in which a user can violate the TSP.

AVA_VLA.4.2C The vulnerability analysis documentation shall describe the disposition of identified potential vulnerabilities.

AVA_VLA.4.3C The vulnerability analysis documentation shall show, for all identified potential vulnerabilities, that the vulnerability cannot be exploited in the operational environment for the TOE.

AVA_VLA.4.4C The vulnerability analysis documentation shall show that the search for potential vulnerabilities is systematic.

AVA_VLA.4.5C The vulnerability analysis documentation shall provide a justification that the analysis completely addresses the TOE deliverables.

Evaluator action elements

AVA_VLA.4.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_VLA.4.2E The evaluator ***shall conduct*** penetration testing, using both the developer vulnerability analysis and the evaluator's independent vulnerability analysis,

to determine that the TOE in its operational environment is resistant to penetration attacks performed by an attacker possessing high attack potential.

A Vulnerability Assessment (AVA) (normative)

458 This annex provides an explanation of the AVA criteria and examples of their application. This annex does not define the AVA criteria, this definition can be found in CC Part 3 Section 13.

459 This annex consists of 4 major parts:

a) *Guidance for completing a Covert Channel Analysis*. This is summarised in section A.1.

b) *Guidance for completing a Misuse Analysis*. This is summarised in section A.2.

c) *What a vulnerability analysis must contain*. This is summarised in section A.3, and described in more detail in sections A.4 to A.5. These sections describe the approach to vulnerability analysis, the contents of the developer Vulnerability Analysis, how an evaluator should approach the construction of an independent Vulnerability Analysis and the interrelationships between these contents.

d) How to characterise and use assumed Attack Potential of an attacker. This is summarised in section A.6, and described in more detail in sections A.7 to A.10. These sections provide an example of describe how an attack potential can be characterised and should be used, and provide examples.

A.1 Guidance for completing a Covert Channel Analysis

460 Channel capacity estimations are based upon informal engineering measurements, as well as actual test measurements.

461 Examples of assumptions upon which the covert channel analysis is based may include processor speed, system or network configuration, memory size, and cache size.

462 The selective validation of the covert channel analysis through testing allows the evaluator the opportunity to verify any aspect of the covert channel analysis (e.g. identification, capacity estimation, elimination, monitoring, and exploitation scenarios). This does not impose a requirement to demonstrate the entire set of covert channel analysis results.

463 If there are no information flow control SFPs in the ST, this family of assurance requirements is no longer applicable, as this family applies only to information flow control SFPs.

464 Performing a covert channel analysis in a systematic way, as required by AVA_CCA.2 Systematic covert channel analysis, requires that the

developer identify covert channels in a structured and repeatable way, as opposed to identifying covert channels in an ad-hoc fashion.

465 Performing a covert channel analysis in an exhaustive way, as required by AVA_CCA.3 Exhaustive covert channel analysis, requires that additional evidence be provided that the plan that was followed for identifying covert channels is sufficient to ensure that all possible ways for covert channel exploration have been exercised.

A.2 Guidance for completing a Misuse Analysis

466 Conflicting, misleading, incomplete or unreasonable guidance may result in a user of the TOE believing that the TOE is secure when it is not, and can result in exploitable vulnerabilities. Therefore, the misuse analysis investigates whether the TOE can be configured or used in a manner that is insecure but that an administrator or user of the TOE would reasonably believe to be secure.

467 An example of conflicting guidance would be two guidance instructions that imply different outcomes when the same input is supplied.

468 An example of misleading guidance would be the description of a single guidance instruction that could be parsed in more than one way, one of which may result in an insecure state.

469 An example of incomplete guidance would be a list of significant physical security requirements that omitted an important item, resulting in this item being overlooked by the administrator who believed the list to be complete.

470 An example of unreasonable guidance would be a recommendation to follow a procedure that imposed an unduly onerous administrative burden.

471 Guidance documentation is required as an input into this analysis. This may be contained in existing User or Administration documentation, or may be provided separately. If provided separately, the evaluator should confirm that the documentation is supplied with the TOE.

472 In AVA_MSU.3 Analysis and testing for insecure states the evaluator is required to undertake testing to ensure that if and when the TOE enters an insecure state this may easily be detected. This testing may be considered as a specific aspect of penetration testing (as performed for Vulnerability analysis (AVA_VLA) components).

A.3 What is Vulnerability Analysis

473 The purpose of the vulnerability assessment activity is to determine the existence and exploitability of flaws or weaknesses in the TOE in the operational environment. This determination is based upon analysis performed by the developer and the evaluator, and is supported by evaluator testing.

474 At the lowest levels of Vulnerability analysis (AVA_VLA) the evaluator simply validates the developer's analysis, while at the higher levels the evaluators performs an independent analysis.

475 There are two main factors in performing a vulnerability analysis, namely;

a) the identification of potential vulnerabilities;

b) penetration testing to determine whether the potential vulnerabilities are exploitable in the operational environment of the TOE.

476 The identification of vulnerabilities can be further decomposed into the evidence to be searched and how hard to search that evidence to identify potential vulnerabilities. In a similar manner, the penetration testing can be further decomposed into analysis of the potential vulnerability to identify attack methods and the demonstration of the attack methods.

477 These main factors are iterative in nature, i.e. penetration testing of potential vulnerabilities may lead to the identification of further potential vulnerabilities. Hence, these are performed as a single vulnerability analysis activity.

A.4 Developer construction of a Vulnerability Analysis

478 A vulnerability analysis is performed by the developer to ascertain the presence of potential vulnerabilities, and should consider at least the contents of all the TOE deliverables including the ST for the targeted evaluation assurance level. The developer is required to document the disposition of identified potential vulnerabilities to allow the evaluator to make use of that information in support of the evaluator's independent vulnerability analysis.

479 The intent of the developer analysis is to confirm that no identified potential vulnerabilities can be exploited in the operational environment for the TOE. This determination is made assuming the role of an attacker possessing a basic (for AVA_VLA.1 Developer vulnerability analysis and AVA_VLA.2 Independent vulnerability analysis), moderate (for AVA_VLA.3 Moderately resistant) or high (for AVA_VLA.4 Highly resistant) attack potential.

480 At AVA_VLA.3 Moderately resistant and AVA_VLA.4 Highly resistant the developer shows a systematic search for potential vulnerabilities has been performed. This systematic search requires that the developer identify those potential vulnerabilities in a structured and repeatable way, as opposed to identifying them in an ad-hoc fashion. The associated evidence that the search for potential vulnerabilities was systematic should include identification of all TOE documentation upon which the search for flaws was based. This should also include any other relevant information considered in the search.

481 The vulnerability analysis for AVA_VLA.4 Highly resistant should include a rationale to demonstrate why the systematic search is considered to be complete.

A.4.1 Unstructured Analysis

482 The unstructured analysis to be performed by the developer (for AVA_VLA.1 Developer vulnerability analysis and AVA_VLA.2 Independent vulnerability analysis) permits the developer to analyse the design of the TOE and provide a justification of how the design protects against attackers. To complete this analysis the developer is able to take a number of different approaches, including:

a) Focusing on aspects of the design that are considered to be particularly complex. For example, the developer will analyse the design to ensure that all routines have been correctly specified and implemented, that all structures are accurately defined and used, that all interfaces have adequate bounds defined to control acceptance of data passed.

b) Analysis of the TOE architecture to determine there are no paths which permit TSFI to be bypassed.

c) Analysis of the TOE architecture to ensure that non-TSF portions can only interact with the TSF through defined TSFI and that interactions within the TSF are only performed through permitted internal interfaces.

d) Consideration of any probabilistic/permutational mechanisms that have an inherent weakness to direct attacks. The developer should ensuring that the interfaces into these mechanisms prevent a brute force attack, such as incremental guesses of the secret TSF data, to be successful.

e) Consideration of experience and knowledge of flaws in similar technology -types.

483 The developer should consider all of the design and operation evidence provided for the evaluation (i.e. that required for any ADV and AGD components included in the assurance package) when performing the analysis of the design.

484 Any of the above may identify a weakness in the TOE construction, which indicates a potential vulnerability. For each potential vulnerability the developer is to provide evidence that this potential vulnerability cannot be exploited in the TOE in its operational environment. The developer may provide details of testing (interface, subsystem or module) to demonstrate that any potential vulnerabilities hypothesised are not present in the TOE or that the TOE is resistant to a particular attack. Or, the developer may refer to another aspect of TOE design that serves to protect against the potential vulnerability.

A.4.2 Systematic analysis

485 The systematic analysis to be performed by the developer (for AVA_VLA.3 Moderately resistant and AVA_VLA.4 Highly resistant) is to be approached in a structured, repeatable manner, such that all necessary inputs are analysed in their entirety. Therefore, where the developer was able to focus attention to specific aspects of the design in the unstructured analysis, the developer has to consider the completeness of the TOE design and construction.

486 In this analysis the developer applies a flaw hypothesis methodology that considers all development and guidance evidence provided for the evaluation in an approach that is complete. To achieve this, a developer may adopt a published flaw hypothesis method or the developer may specify their own methodology that is applied. If the developer chooses to specify their own methodology, the method should include all approaches identified in the unstructured analysis, together with consideration of how the TOE architecture has been specified and developed to ensure its completeness in controlling all interfaces through which an attacker may attempt to access the TOE.

487 A rationale for the completeness of the method applied must be provided for AVA_VLA.4 Highly resistant, to demonstrate that the entire TOE design has been analysed. This does not imposed any additional requirements upon the analysis to be performed, merely that a rationale is to be provided.

A.5 Evaluator construction of a Vulnerability Analysis

488 Independent evaluator vulnerability analysis goes beyond the potential vulnerabilities identified by the developer. The evaluator analysis is to determine that the TOE is resistant to penetration attacks performed by an attacker possessing a basic (for AVA_VLA.2 Independent vulnerability analysis), moderate (for AVA_VLA.3 Moderately resistant) or high (for AVA_VLA.4 Highly resistant) attack potential. The evaluator first assesses the exploitability of all identified potential vulnerabilities. This is accomplished by conducting penetration testing. The evaluator should assume the role of an attacker with a basic (for AVA_VLA.2 Independent vulnerability analysis), moderate (for AVA_VLA.3 Moderately resistant) or high (for AVA_VLA.4 Highly resistant) attack potential when attempting to penetrate the TOE.

489 The evaluator considers potential vulnerabilities encountered by the evaluator during the conduct of other evaluation activities. The evaluator penetration testing determining TOE resistance to these potential vulnerabilities should be performed assuming the role of an attacker with a basic (for AVA_VLA.1 Developer vulnerability analysis and AVA_VLA.2 Independent vulnerability analysis), moderate (for AVA_VLA.3 Moderately resistant) or high (for AVA_VLA.4 Highly resistant) attack potential.

490 However, vulnerability analysis should not be performed as an isolated activity. It is closely linked with ADV: Development and AGD: Guidance documents. The evaluator performs these other evaluation activities with a focus on identifying potential vulnerabilities or "areas of concern". Therefore, evaluator familiarity with the generic vulnerability guidance (provided in CEM, AVA_VLA.2 Independent vulnerability analysis-5).

A.6 Identification of Potential Vulnerabilities

491 Potential vulnerabilities may be identified by the evaluator during different activities. They may become apparent during an evaluation activity or they may be identified as a result of analysis of evidence to search for vulnerabilities.

A.6.1 Encountered

492 The encountered identification of vulnerabilities is where potential vulnerabilities are identified by the evaluator during the conduct of evaluation activities, i.e. the evidence are not being analysed with the express aim of identifying potential vulnerabilities.

493 The encountered method of identification is dependent on the evaluator's experience and knowledge; which is monitored and controlled by the Certification Authority. It is not reproducible in approach, but will be documented to ensure repeatability of the conclusions from the reported potential vulnerabilities.

494 There is no formal analysis criteria required for this method. Potential vulnerabilities are identified from the evidence provided as a result of knowledge and experience. However, this method of identification is not constrained to any particular subset of evidence.

495 Evaluator is assumed to have knowledge of the TOE-type technology and known security flaws as documented in the public domain. The level of knowledge assumed is that which can be gained from a security e-mail list relevant to the TOE type, the regular bulletins (bug, vulnerability and security flaw lists) published by those organisations researching security issues in products and technologies in widespread use. This knowledge is not expected to extend to specific conference proceedings or detailed theses produced by university research. However, to ensure the knowledge applied is up to date, the evaluator may need to perform a search of public domain material.

496 Examples of how these may arise (how the evaluator may encounter potential vulnerabilities):

a) while the evaluator is examining some evidence, it sparks a memory of a potential vulnerability identified in a similar product type, that the evaluator believes to also be present in the TOE under evaluation;

b) while examining some evidence, the evaluator spots a flaw in the specification of an interface, that reflects a potential vulnerability.

497 This may include becoming aware of a potential vulnerability in a TOE through reading about generic vulnerabilities in a particular product type in an IT security publication or on a security e-mail list to which the evaluator is subscribed.

498 Attack methods can be developed directly from these potential vulnerabilities. Therefore, the encountered potential vulnerabilities are collated at the time of producing penetration tests based on the developer's vulnerability analysis or, if performed, an independent vulnerability analysis. There is no explicit action for the evaluator to encounter potential vulnerabilities. Therefore, the evaluator is directed through an implicit action specified in Vulnerability analysis (AVA_VLA).*.2E.

499 Current information regarding public domain vulnerabilities and attacks may be provided to the evaluator by, for example, an overseer such as the evaluation authority. This information is to be taken into account by the evaluator when collating encountered vulnerabilities and attack methods when developing penetration tests.

A.6.2 Analysis

500 The following types of analysis are presented in terms of the evaluator actions. However, the general philosophy and approaches described are also applicable to the developer in the production of the evaluation evidence.

A.6.2.1 Focused

501 During the conduct of evaluation activities the evaluator may also identify areas of concern. These are specific portions of the TOE evidence that the evaluator has some reservation about, although the evidence meets the requirements for the activity with which the evidence is associated. For example, a particular interface specification looks particularly complex, and therefore may be prone to error either in the construction of the TOE or in the operation of the TOE. There is no potential vulnerability apparent at this stage, further investigation is required. This is beyond the bounds of encountered, as further investigation is required.

502 Difference between potential vulnerability and area of concern:

a) Potential vulnerability - know a method of attack that can be used to exploit it, know of vulnerability information.

b) Area of concern - may be able to discount concern as a potential vulnerability based on information provided elsewhere. While reading interface specification, the evaluator identifies that due to the extreme (unnecessary) complexity of an interface a potential vulnerability may lie within that area, although it is not apparent through this initial examination.

503 The focused approach to the identification of vulnerabilities is an analysis of the evidence with the aim of identifying any potential vulnerabilities evident through the contained information. It is an unstructured analysis, as the approach is not predetermined. This approach to the identification of potential vulnerabilities can be used during the independent vulnerability analysis required by AVA_VLA.2 Independent vulnerability analysis.

504 This analysis can be achieved through different approaches, that will lead to commensurate levels of confidence. None of the approaches have a rigid format for the examination of evidence to be performed.

505 The approach taken is directed by the results of the evaluator's assessment of the evidence to determine it meets the requirements of the ADV/AGD sub-activities. Therefore, the investigation of the evidence for the existence of potential vulnerabilities may be directed by any of the following:

a) areas of concern identified during examination of the evidence during the conduct of evaluation activities;

b) directive from developer vulnerability analysis, leading the evaluator to examine particular areas of the TOE evidence;

c) representative examination of the evidence to hypothesise potential vulnerabilities in the TOE.

506 The evaluator will report what actions were taken to identify potential vulnerabilities in the evidence. However, the evaluator may not be able to describe the steps in identifying potential vulnerabilities before the outset of the examination. The approach will evolve as a result of the outcome of evaluation activities.

507 The areas of concern may arise from examination of any of the evidence provided to satisfy the SARs specified for the TOE evaluation. The information publicly accessible is also considered.

508 The activities performed by the evaluator can be repeated and the same conclusions, in terms of the level of assurance in the TOE, can be reached although the steps taken to achieve those conclusions may vary. As the evaluator is documenting the form the analysis took, the actual steps taken to achieve those conclusions are also reproducible.

A.6.2.2 Methodical

509 The methodical analysis approach takes the form of a structured examination of the evidence. This method requires the evaluator to specify the structure and form the analysis will take (i.e. the manner in which the analysis is performed is predetermined, unlike the focused identification method). The method is specified in terms of the information that will be considered and how/why it will be considered. This approach to the identification of potential vulnerabilities can be used during the independent vulnerability

analysis required by AVA_VLA.3 Moderately resistant and AVA_VLA.4 Highly resistant.

510 This analysis of the evidence is deliberate and pre-planned in approach, considering all evidence identified as an input into the analysis.

511 All evidence provided to satisfy the (ADV) assurance requirements specified in the assurance package are used as input to the potential vulnerability identification activity.

512 The "methodical" descriptor for this analysis has been used in an attempt to capture the characterisation that this identification of potential vulnerabilities is to take an ordered and planned approach. A "method" or "system" is to be applied in the examination. The evaluator is to describe the method to be used in terms of what evidence will be considered, the information within the evidence that is to be examined, the manner in which this information is to be considered; and the hypothesis that is to be generated.

513 The following provide some examples that a hypothesis may take:

a) consideration of malformed input for interfaces available to an attacker at the external interfaces;

b) examination of a security mechanism, such as domain separation, hypothesising internal buffer overflows leading to degradation of separation;

c) analysis to identify any objects created in the TOE implementation representation that are then not fully controlled by the TSF, and could be used by an attacker to undermine the TSP.

514 For example, the evaluator may identify that interfaces are a potential area of weakness in the TOE and specify an approach to the analysis that "all interface specifications provided in the functional specification and high-level design will be analysed to hypothesise potential vulnerabilities" and go on to explain the methods used in the hypothesis.

515 This identification method will provide a plan of attack of the TOE, that would be performed by an evaluator completing penetration testing of potential vulnerabilities in the TOE. The rationale for the method of identification would provide the evidence for the coverage and depth of exploitation determination that would be performed on the TOE.

A.7 When is attack potential used

A.7.1 Developer

516 Attack potential is used by a PP/ST author during the development of the PP/ST, in consideration of the threat environment and the selection assurance components. This may simply be a determination that the attack potential possessed by the assumed attackers of the TOE is generically characterised

as basic, moderate or high. Alternatively, the PP/ST may wish to specify particular levels of individual factors assumed to be possessed by attackers. (e.g. the attackers are assumed to be experts in the TOE technology type, with access to specialised equipment.)

517 The PP/ST author considers the threat profile developed during a risk assessment (outside the scope of the CC, but used as an input into the development of the PP/ST in terms of the Security Problem Definition). Consideration of this threat profile in terms of one of the approaches discussed in the following sections will permit the specification of the attack potential the TOE is to resist.

A.7.2 Evaluator

518 Attack potential is especially considered by the evaluator in two distinct ways during the ST evaluation and the vulnerability assessment activities.

519 During the ST evaluation, the evaluator determines whether or not the choice of the assurance requirement components, in particular the components of the AVA: Vulnerability assessment class, are commensurate with the threat attack potential (see ASE_REQ.1.4C). Cases where the assurance is not commensurate may mean either that the evaluation will not provide sufficient assurance, or that the evaluation will be unnecessarily onerous.

520 Attack potential is used by an evaluator during the conduct of the vulnerability analysis sub-activity to determine whether or not the TOE is resistant to attacks assuming a specific attack potential of an attacker. If the evaluator determines that a potential vulnerability is exploitable in the TOE, they have to confirm that it is exploitable considering all aspects of the intended environment, including the attack potential assumed by an attacker.

521 Therefore, using the information provided in the threat statement of the Security Target, the evaluator determines the minimum attack potential required by an attacker to effect an attack, and arrives at some conclusion about the TOE's resistance to attacks. Table 10 Vulnerability testing and attack potential demonstrates the relationship between this analysis and attack potential.

vulnerability component	TOE resistant to attacker with attack potential of:	Residual vulnerabilities only exploitable by attacker with attack potential of:
VLA.4	high	*infeasible*
VLA.3	moderate	high
VLA.2	basic	moderate
VLA.1	basic	moderate

Table 10 Vulnerability testing and attack potential

522 The “infeasible” attack potential in the residual vulnerabilities column of the above table represents those potential vulnerabilities that would become exploitable should a countermeasure in the operational environment be

removed or the operational environment develop such that the technology required to perform an exploit becomes more widely available.A vulnerability classified as residual in this instance reflects the fact that a known weakness exists in the TOE, but in the current operational environment, with the assumed attack potential, the weakness cannot be exploited.

523 A vulnerability analysis applies to all TSFI, including ones that access probabilistic or permutational mechanisms. No assumptions are made regarding the correctness of the design and implementation of the TSFI; nor are constraints placed on the attack method or the attacker's interaction with the TOE - if an attack is possible, then it is to be considered during the vulnerability analysis. As shown in Table 10 Vulnerability testing and attack potential , successful evaluation against a vulnerability assurance component reflects that the TSF is designed and implemented to protect against the required level of threat.

524 It is not necessary for an evaluator to perform an attack potential calculation for each potential vulnerability. In some cases it is apparent when developing the attack method whether or not the attack potential required to develop and run the attack method is commensurate with that assumed of the attacker in the operational environment. For any vulnerabilities for which an exploitation is determined, the evaluator performs an attack potential calculation to determine that the exploitation is appropriate to the level of attack potential assumed for the attacker.

525 This material is not intended to select and specify a preferred method. Rather it is to provide alternatives for PP/ST authors and evaluators to consider when assuming a level of attack potential.

A.8 Weighted parameters Approach

A.8.1 Application of attack potential

526 Attack potential is a function of expertise, resources and motivation. There are multiple methods of representing and quantifying these factors. Also, there are other factors that are applicable for particular TOE types. The following material presents one method.

A.8.1.1 Treatment of motivation

527 Motivation is an attack potential factor that can be used to describe several aspects related to the attacker and the assets the attacker desires. Firstly, motivation can imply the likelihood of an attack - one can infer from a threat described as highly motivated that an attack is imminent, or that no attack is anticipated from an un-motivated threat. However, except for the two extreme levels of motivation, it is difficult to derive a probability of an attack occurring from motivation.

528 Secondly, motivation can imply the value of the asset, monetarily or otherwise, to either the attacker or the asset holder. An asset of very high

value is more likely to motivate an attack compared to an asset of little value. However, other than in a very general way, it is difficult to relate asset value to motivation because the value of an asset is subjective - it depends largely upon the value an asset holder places on it.

529 Thirdly, motivation can imply the expertise and resources with which an attacker is willing to effect an attack. One can infer that a highly motivated attacker is likely to acquire sufficient expertise and resources to defeat the measures protecting an asset. Conversely, one can infer that an attacker with significant expertise and resources is not willing to effect an attack using them if the attacker's motivation is low.

530 During the course of preparing for and conducting an evaluation, all three aspects of motivation are at some point considered. The first aspect, likelihood of attack, is what may inspire a developer to pursue an evaluation. If the developer believes that the attackers are sufficiently motivated to mount an attack, then an evaluation can provide assurance of the ability of the TOE to thwart the attacker's efforts. Where the operational environment is well defined, for example in a system evaluation, the level of motivation for an attack may be known, and will influence the selection of countermeasures.

531 Considering the second aspect, an asset holder may believe that the value of the assets (however measured) is sufficient to motivate attack against them. Once an evaluation is deemed necessary, the attacker's motivation is considered to determine the methods of attack that may be attempted, as well as the expertise and resources used in those attacks. Once examined, the developer is able to choose the appropriate assurance level, in particular the AVA: Vulnerability assessment requirement components, commensurate with the attack potential for the threats. During the course of the evaluation, and in particular as a result of completing the vulnerability assessment activity, the evaluator determines whether or not the TOE, operating in its operational environment, is sufficient to thwart attackers with the identified expertise and resources.

532 It may be possible for a PP author to quantify the motivation of an attacker, as the PP author has greater knowledge of the operational environment in which the TOE (conforming to the requirements of the PP) is to be placed. Therefore, the motivation could form an explicit part of the expression of the attack potential in the PP, along with the necessary methods and measures to quantify the motivation.

A.8.2 Characterising attack potential

533 This section examines the factors that determine attack potential, and provides some guidelines to help remove some of the subjectivity from this aspect of the evaluation process.

A.8.2.1 Identification and exploitation

534 For an attacker to exploit a vulnerability in the TOE, the potential vulnerability must first be identified, the attack method then developed and finally the potential vulnerability exploited using the attack method. Each of these stages of determining whether there is a vulnerability in the TOE must be considered when quantifying the factors comprising the attack potential.

535 To illustrate this, consider a potential vulnerability that is uncovered following months of analysis by an expert, but requires use of a simple attack method published on the Internet to exploit. Compare this with a potential vulnerability that is well known, but requires enormous time and resource to exploit.

536 When a vulnerability is identified by an evaluator, the evaluator must determined the attack potential associated with the vulnerability. The evaluator may have performed considerable analysis to identify the vulnerability. However, the evaluator must consider the effect of the vulnerability becoming publicly known. That is, an attacker would not have to repeat the analysis to identify the vulnerability, but would only have to perform the exploitation. In some instances knowledge of the vulnerability would not immediately facilitate exploitation because considerable further analysis would be required to permit the development of an attack method

537 In direct attacks against probabilistic or permutational mechanisms, the issue of exploitation will normally be the most important, since potential vulnerabilities in these mechanisms will often be self evident. Note, however, that this may not always be the case. With cryptographic mechanisms, for example, knowledge of subtle potential vulnerabilities may considerably affect the effectiveness of a brute force attack. Knowledge that users of a system tend to choose first names as passwords will have a similar effect. For vulnerability testing above AVA_VLA.1 Developer vulnerability analysis, the initial identification of potential vulnerabilities will become a much more important consideration, since the existence of difficult to uncover potential vulnerabilities may be promulgated, often rendering exploitation trivial.

A.8.2.2 Factors to be considered

538 The following factors should be considered during analysis of the attack potential required to exploit a vulnerability:

a) Time taken to identify and exploit (***Elapsed Time***);

b) Specialist technical expertise required (***Specialist Expertise***);

c) Knowledge of the TOE design and operation (***Knowledge of the TOE***);

d) ***Window of opportunity***;

e) ***IT hardware/software or other equipment*** required for exploitation.

539 These factors provide characterisation of other quantifiers that can be used to describe the attack potential posed by an attacker, such as motivation and collusion.

540 Motivation and the value of the asset are intrinsically linked, as they give an indication of the lengths to which an attacker will go in order to subvert the TSP of the TOE. If the asset is of high value to the attacker, either in monetary value or in prestige of attaining possession, the attacker is likely to have a high motivation and will sustain his efforts in subverting the TOE. This may be demonstrated by seeking to increase his access to the required knowledge of the TOE, related technology and/or attack methods (either by increasing his own or looking to external sources), increasing the sophistication of the equipment available for the attack, and by dedicating a large amount of time to the attack. The ability to increase these factors and sustain the effort applied to the attack is likely to depend upon the funds at the attacker's disposal, as many of the factors (***Knowledge of the TOE***, ***Equipment***, and even ***Elapsed Time*** to a certain extent) can be purchased by the attacker.

541 The attacker may also seek to increase the factors by colluding with others. For this reason the attack potential is calculated as that possessed by the combination of the people involved in an attack, providing a characterisation of the role of the attacker. Therefore, the different types of expertise required at each stage of the attack, and within each stage of the attack, must be considered. Different levels of expertise may be required between the identification of the potential vulnerability, the development of the attack method and the realisation of the attack. Within each of these stages a number of different types of expertise may be required. Therefore the highest level of required expertise must be assumed when applying.

542 In many cases these factors are not independent, but may be substituted for each other in varying degrees. For example, expertise or hardware/software may be a substitute for time. A discussion of these factors follows. (The levels of each factor are discussed in increasing order of magnitude.)

543 Elapsed time is the total amount of time taken by an attacker to identify that a particular potential vulnerability may exist in the TOE, to develop an attack method and to sustain effort required to mount the attack against the TOE. When considering this factor, the worst case scenario should be used to estimate the amount of time required.

544 For example, the time taken to identify a potential vulnerability may be the time taken to locate the potential vulnerability in the information that is publicly available or may be the time required to analyse the design information to identify a potential vulnerability. In addition to this time taken for identification, consideration of the time required to develop an attack method (which may also be publicly available) and successfully run the attack method on the TOE to exploit the vulnerability must be included in the ***Elapsed Time*** factor.

545 For the purposes of this discussion within minutes means an attack can be identified or exploited in less than an hour; within hours means an attack can succeed in less than a day; within days means an attack can succeed in less than a week, within weeks means an attack can succeed in less than a month, and in months means a successful attack requires up to three months.

546 Specialist expertise refers to the level of generic knowledge of the underlying principles, product type or attack methods (e.g. Internet protocols, Unix operating systems, buffer overflows). The identified levels are as follows:

a) Laymen are unknowledgeable compared to experts or proficient persons, with no particular expertise;

b) Proficient persons are knowledgeable in that they are familiar with the security behaviour of the product or system type;

c) Experts are familiar with the underlying algorithms, protocols, hardware, structures, security behaviour, principles and concepts of security employed, techniques and tools for the definition of new attacks, cryptography, classical attacks for the product type, attack methods, etc. implemented in the product or system type.

547 When describing the expertise required, the total number of experts required must be included; the number of people for each type of expertise required and access to the expertise (dissemination) must be considered when describing the expertise required. Therefore, if expertise in both techniques for types of attack applicable to the TOE and underlying algorithms and protocols is required, then the highest level of ***Specialist Expertise*** characterisation should be assumed.

548 ***Knowledge of the TOE*** refers to specific expertise in relation to the TOE. This is distinct from generic expertise, but not unrelated to it. Identified levels are as follows:

a) Public information concerning the TOE (e.g. as gained from the internet);

b) Restricted information concerning the TOE (e.g. knowledge that is controlled within the developer organisation and shared with other organisations under a non-disclosure agreement)

c) Sensitive information about the TOE (e.g. knowledge that is shared between discreet teams within the developer organisation, access to which is constrained only to members of the specified teams);

d) Critical information about the TOE (e.g. knowledge that is known by only a few individuals, access to which is very tightly controlled on a strict need to know basis and individual undertaking).

549 The knowledge of the TOE may graduate according to design abstraction, although this can only be done on a TOE by TOE basis. Some TOE designs

may be public source (or heavily based on public source) and therefore even the design representation would be classified as public or at most restricted, while the implementation representation for other TOEs is very closely controlled as it would give an attacker information that would aid an attack and is therefore considered to be sensitive or even critical.

550 Care should be taken here to ensure the highest level of knowledge of the TOE required during identification, development and running of the potential vulnerability is identified.

551 ***Window of opportunity*** (Opportunity) is also an important consideration, and has a relationship to the ***Elapsed Time*** factor. Identification or exploitation of a vulnerability may require considerable amounts of access to a TOE that may increase the likelihood of detection. Some attack methods may require considerable effort off-line, and only brief access to the TOE to exploit. Access may also need to be continuous, or over a number of sessions.

552 For some TOEs the ***Window of opportunity*** may equate to the number of samples of the TOE that the attacker can obtain. This is particularly relevant where attempts to penetrate the TOE and undermine the TSP may result in the destruction of the TOE preventing use of that TOE sample for further testing, e.g.hardware devices. Often in these cases distribution of the TOE is controlled and so the attacker must apply effort to obtain further samples of the TOE.

553 For the purposes of this discussion unnecessary/unlimited access means that the attack doesn”t need any kind of opportunity to be realised; easy means that access is required for less than a day or that the number of TOE samples required to perform the attack is less than ten; moderate means that access is required for less than a month or that the number of TOE samples required to perform the attack is less than fifty; difficult means that access is required for at least a month or that the number of TOE samples required to perform the attack is less than one hundred; none means that the opportunity window is not sufficient to perform the attack (the length for which the asset to be exploited is available or is sensitive is less than the opportunity length needed to perform the attack - for example, if the asset key is changed each week and the attack needs two weeks).

554 Consideration of this factor may result in a determining that it is not possible to complete the exploit, due to requirements for time availability that are greater than the opportunity time.

555 ***IT hardware/software or other equipment*** refers to the equipment required to identify or exploit a vulnerability.

a) Standard equipment is readily available to the attacker, either for the identification of a vulnerability or for an attack. This equipment may be a part of the TOE itself (e.g. a debugger in an operating system), or can be readily obtained (e.g. Internet downloads, protocol analyser or simple attack scripts).

b) Specialised equipment is not readily available to the attacker, but could be acquired without undue effort. This could include purchase of moderate amounts of equipment (e.g. power analysis tools, use of hundreds of PCs linked across the Internet would fall into this category), or development of more extensive attack scripts or programs.

c) Bespoke equipment is not readily available to the public as it may need to be specially produced (e.g. very sophisticated software), or because the equipment is so specialised that its distribution is controlled, possibly even restricted. Alternatively, the equipment may be very expensive.

556 Specialist expertise and ***Knowledge of the TOE*** are concerned with the information required for persons to be able to attack a TOE. There is an implicit relationship between an attacker's expertise (where the attacker may be one or more persons with complementary areas of knowledge) and the ability to effectively make use of equipment in an attack. The weaker the attacker's expertise, the lower the potential to use equipment (IT hardware/software or other equipment). Likewise, the greater the expertise, the greater the potential for equipment to be used in the attack. Although implicit, this relationship between expertise and the use of equipment does not always apply, for instance, when environmental measures prevent an expert attacker's use of equipment, or when, through the efforts of others, attack tools requiring little expertise to be effectively used are created and freely distributed (e.g. via the Internet).

A.8.2.3 An approach to calculation

557 The above section identifies the factors to be considered. However, further guidance is required if evaluations are to be conducted on a consistent basis. The following approach is provided to assist in this process. The numbers have been provided with the objective of achieving ratings that are consistent with the relevant evaluation levels.

558 Table 11 Calculation of attack potential identifies the factors discussed in the previous section and associates numeric values with the total value of each factor.

559 When this table is used by a PP/ST author the highest level of each factor assumed to be applied by an attacker is to be identified. The values associated with these levels are then identified using Table 11 Calculation of attack potential, and are summed to determine the overall attack potential rating of the assumed attacker. This should be performed in the context of threats against the TOE as specified in the Security Problem Definition of the ST.

560 When determining the attack potential for a given vulnerability, a single level for each factor is selected to represent the extent of the factor required to identify the potential vulnerability, develop an attack method and perform the exploitation. The selected level should reflect the highest level of the

factor required. Although in the case of Expertise if more than one type of expert is required, this factor should be iterated (see below). The values associated with the selected level for each factor should then be identified using Table 11 Calculation of attack potential. When selecting values the intended operational environment for the TOE should be assumed. The values are then summed, giving a single value. This value is then checked using Table 12 Rating of vulnerabilities to determine the overall rating.

561 If the attacker needs to have different types of expertise (example: hardware specialist and firewall expert) or different types of equipment (example: protocol analyser and very sophisticated software) to develop and perform the attack, then the different values corresponding to the factor (***Specialist Expertise*** or ***Equipment***) should be added. If you only need one type of expertise or equipment, no iteration should be performed. Elapsed time, Knowledge of the TOE and Window of Opportunity factors cannot be iterated: only the overall sequence of steps to perform the attack should be considered (identification of a potential vulnerability if any, development of attack method if any and realisation of the attack).

562 For the ***Elapsed Time*** factor, each week is considered to be worth one point, so this factor can scale in the granularity required for the TOE.

563 Where a factor falls close to the boundary of a range the evaluator should consider use of an intermediate value to those in the table. For example, if twenty samples are required to perform the attack then a value between one and four may be selected for that factor, or if the design is based on a publicly available design but the developer has made some alterations then a value between zero and four should be selected according to the evaluator's view of the impact of those design changes. The table is intended as a guide.

564 The "**" specifications in the table are not to be seen as a natural progression from the timescales specified in the preceding ranges associated with a factor. These specifications identify that for a particular reason the potential vulnerability cannot be exploited in the TOE in its intended operational environment. For example, in considering ***Window of Opportunity***, unauthorised access to the TOE may be detected after a certain amount of time in a TOE with a known environment (i.e. in the case of a system) where regular patrols are completed, and the attacker could not gain access to the TOE for the required two weeks undetected. However, this would not be applicable to a TOE connected to the network where remote access is possible, or where the physical environment of the TOE is unknown.

Factor	Range	Value
Elapsed Time	<= 1 day	0
	<= 1 week	1
	<= 1 month	4
	<= 3 months	13
	<= 6 months	26
	> 6 months	*

Factor	Range	Value
Expertise	Layman	0
	Proficient	2
	Expert	5
Knowledge of TOE	Public	0
	Restricted	1
	Sensitive	4
	Critical	10
Window of Opportunity	Unnecessary / unlimited access	0
	Easy	1
	Moderate	4
	Difficult	12
	None	**
Equipment	Standard	0
	Specialised	3
	Bespoke	7

Table 11 Calculation of attack potential

565 * Indicates that the corresponding attack potential is beyond high attack potential.

566 ** Indicates that the attack path is not exploitable due to other measures in the intended operational environment of the TOE.

567 For a given vulnerability it may be necessary to make several passes through the table for different attack scenarios (e.g. trading off, or compensating, expertise for time or equipment). The lowest value obtained for these passes should be retained, as this reflects the minimum level of attack potential required to undermine the TSP.

568 In the case of a vulnerability that has been identified and is in the public domain, the identifying values should be selected for an attacker to uncover that vulnerability in the public domain, rather than to initially identify it.

569 If different types of attacker are assumed by the PP/ST author, several passes through the table should be made to determine the different level of attack potential understood for each type of attacker. The PP/ST author then considers the highest value obtained when determining the level of attack the TOE should withstand (selection of Vulnerability analysis (AVA_VLA) component).

570 Table 12 Rating of vulnerabilities should then be used to obtain a rating for the vulnerability/attack potential.

Range of values	Resistant to attacker with attack potential of:
0-2	No rating
3-6	Basic
7-14	Moderate
15-26	High
*	Beyond High

Table 12 Rating of vulnerabilities

571 An approach such as this cannot take account of every circumstance or factor, but should give a better indication of the level of resistance to attack required to achieve the standard ratings. Other factors, such as the reliance on unlikely chance occurrences are not included in the basic model, but can be used by an evaluator as justification for a rating other than those that the basic model might indicate.

572 It should be noted that whereas a number of vulnerabilities rated individually may indicate high resistance to attack, collectively the combination of vulnerabilities may indicate that overall a lower rating is applicable. The presence of one vulnerability may make another easier to exploit.

A.8.3 Examples of the application of this approach

A.8.3.1 Basic attack potential

573 The characterisation of the least attributes required by an attacker demonstrating a Basic attack potential rating is considered to be represented by the following, giving a result of 4 from Table 11 Calculation of attack potential, and a rating of Basic in Table 12 Rating of vulnerabilities :

– the TOE would withstand attack for up to 4 weeks (4);

– layman expertise (0);

– public knowledge of the TOE (0);

– unlimited access/unlimited number of samples (0);

– standard equipment (0).

A.8.3.2 Considering Elapsed Time only

574 The following examples consider a change only in the elapsed time taken to exploit a vulnerability in the TOE, showing the affect on the rating of attack potential for a layman, with standard equipment, public knowledge of the TOE and unlimited access/unlimited number of samples:

a) The TOE can be broken within one week (1) = No rating.

b) The TOE can withstand the attack for up to 4 weeks (4) = Basic.

c) The TOE can withstand the attack for up to 12 weeks (12) = Moderate.

d) The TOE can withstand the attack for more than 13 weeks (13+) = (High).

A.8.3.3 Comparing time and expertise

575 The following examples illustrate the affect on the rating of attack potential when changing the expertise of an attacker and the elapsed time taken to exploit a vulnerability in the TOE. Standard equipment, public knowledge of the TOE and unlimited access/unlimited number of samples are assumed to be used in all cases.

a) The proficient attacker (2) takes between one day and one week (1) to break the TOE, with no other factors = Basic rating (3).

b) The proficient attacker (2) takes up to four weeks (one month) to break the TOE (4), with no other factors = Moderate rating (6).

c) The proficient attacker (2) takes up to twelve weeks (three months) to break the TOE (12), with no other factors = High rating (14).

d) The expert attacker (5) takes between one day and one week (1) to break the TOE, with no other factors = Basic rating (6).

e) The expert attacker (5) takes up to four weeks (one month) to break the TOE (4), with no other factors = Moderate rating (9).

f) The expert attacker (5) takes up to twelve weeks (three months) to break the TOE (12), with no other factors = High rating (17).

A.8.3.4 Elapsed time, Specialist expertise, Knowledge of the TOE

576 The following provides examples of how ratings vary according to the elapsed time, expertise and knowledge that is applied in the attack, when standard equipment is used with unlimited access/unlimited number of samples:

a) TOE resists attack from proficient attacker (2), with restricted knowledge (1), 1 day to 1 week (1) = Basic rating (4).

b) TOE resists attack from laymen with critical information for at least one week = High (14).

c) TOE resists attack from a layman with critical information between one day and one week = Moderate (11).

This example reflects a calculation performed by an evaluator when a vulnerability has been found to result in a successful attack and the evaluator is to rate it as “exploitable” or “residual”.

d) The TOE can be broken by a layman with critical information in less than one day = Moderate (10).

A.9 Example calculation for direct attack

577 Mechanisms subject to direct attack are often vital for system security and developers often strengthen these mechanisms. As an example, a TOE might use a simple pass number authentication mechanism that can be overcome by an attacker who has the opportunity to repeatedly guess another user's pass number. The system can strengthen this mechanism by restricting pass numbers and their use in various ways. During the course of the evaluation an analysis of this direct attack could proceed as follows:

578 Information gleaned from the ST and design evidence reveals that identification and authentication provides the basis upon which to control access to network resources from widely distributed terminals. Physical access to the terminals is not controlled by any effective means. The duration of access to a terminal is not controlled by any effective means. Authorised users of the system choose their own pass numbers when initially authorized to use the system, and thereafter upon user request. The system places the following restrictions on the pass numbers selected by the user:

a) the pass number must be at least four and no greater than six digits long;

b) consecutive numerical sequences are disallowed (such as 7,6,5,4,3);

c) repeating digits is disallowed (each digit must be unique).

579 Guidance provided to the users at the time of pass number selection is that pass numbers should be as random as possible and should not be affiliated with the user in some way - a date of birth, for instance.

580 The pass number space is calculated as follows:

a) Patterns of human usage are an important considerations that can influence the approach to searching a password space. Assuming the worst case scenario and the user chooses a number comprising only four digits, the number of pass number permutations assuming that each digit must be unique is:

$$7(8)(9)(10)? = 5040$$

b) The number of possible increasing sequences is seven, as is the number of decreasing sequences. The pass number space after disallowing sequences is:

$$5040 - 14 = 5026$$

581 Based on further information gleaned from the design evidence, the pass number mechanism is designed with a terminal locking feature. Upon the sixth failed authentication attempt the terminal is locked for one hour. The failed authentication count is reset after five minutes so that an attacker can at best attempt five pass number entries every five minutes, or 60 pass number entries every hour.

582 On average, an attacker would have to enter 2513 pass numbers, over 2513 minutes, before entering the correct pass number. The average successful attack would, as a result, occur in slightly less than:

$$7(8)(9)(10)? = 5040$$

583 Using the approach described in the previous section, it is possible that a layman can defeat the mechanism within days (given easy access to the TOE), with the use of standard equipment, and with no knowledge of the TOE, giving a value of 2. Given the resulting sum, 2, the attack potential required to effect a successful attack is not rated, as it falls below that considered to be Basic.

A.10 Independent Factors Approach

584 This approach is a variation of the Weighted Factors approach presented above. Rather than providing weighting of factors and taking the sum of the weights into account to derive the attack potential, this approach presents a set of independent parameters that require no weighting when assigning initial values to the parameters. (However, there is a sense of implicit weighting as minimum values for a given rating are set for each factor.)

A.10.1 Definitions of Independent Attack Potential Parameters

585 In consideration of the “Nature of the TOE” and the “Life expectancy of the TOE” the following independent parameters have been identified, which affect an estimation of the resistance of a potential vulnerability against a specific attack.

586 The “Nature of the TOE” is the term used to refer to the TOE type, the complexity of the technology used in the construction of the TOE and the information relating to the technology used in the TOE that is available in the public domain. Each of these three aspects affect the level of understanding of the “Nature of the TOE”.

587 The “Life expectancy of the TOE” is used in consideration of the time taken to develop and run an attack method to undermine the TSP. This may be affected by aspects such as the frequency with which a key or password is updated, and the duration in which the TOE is expected to be in operational use.

A.10.1.1 Knowledge of the Technology

588 *The kind of theoretical knowledge necessary to identify a potential vulnerability within the life expectancy of the TOE.*

589 The main focus of this parameter is the technology on which the TOE is based. It strongly depends on the nature of the TOE. This means that the technology of a limited (simple) software product is considered to need less time to become familiar with than with a complex product like an ICC or an operating system. Therefore, identifying potential vulnerabilities depends on both the nature of the TOE and the expertise of the attacker.

590 The following levels of knowledge (expertise) of a special technology have been defined: *Inexperienced-Layman*, *Low-Experienced-Layman*, *Proficient Attacker* or *Expert*.

– An *Inexperienced-Layman* is defined as someone who does not have any useful knowledge of the technology in the subject area of the TOE whereas a *Low-Experienced-Layman* has little knowledge that can be seen as useful.

– A *Proficient Attacker* has at least intermediate knowledge of the technique. Meaning the attacker is familiar with general theory on which the TOE based in respect to its security behaviour.

– An *Expert* has the extensive and profound knowledge required to identify a potential vulnerability in accordance with the nature of the TOE. This not only includes general knowledge of the theory on which the TOE is based but also detailed knowledge on which the theory itself is based. These could be algorithms, tools, the internal structure of the theory etc.

591 There are two possibilities to obtain knowledge of a potential vulnerability. Firstly to study the technology on which the TOE is based. Secondly to have a well known list that identifies, and possibly explains, certain vulnerabilities.

592 In the first case, this parameter depends on the time and/or the expertise an attacker needs to gain knowledge of the vulnerability. The second case assumes that someone already has information of the potential vulnerability. Then the situation should be interpreted as a worst case, whereby the knowledge of the attacker should be estimated as an in-experienced-layman, as the attacker has been given all the knowledge required. The knowledge required to effect an attack on the TOE having gained understanding of the existence of the potential vulnerability is considered in Knowledge of Exploitation, below.

A.10.1.2 Knowledge of the TOE

593 *The kind of knowledge, or possibility to obtain this knowledge, an attacker has or can obtain within the life expectancy of the TOE.*

594 If a potential vulnerability is based on a special knowledge of the TOE (internals) this parameter indicates the depth of information an attacker requires. An attacker may be able to identify potential vulnerabilities in the TOE from TOE design information.

595 The following levels of ***Knowledge of the TOE*** or parts of it have been defined: *None*, *Restricted*, *Sensitive* or *Critical*.

– This parameter is determined to be "*None*" if an attacker has no or only very little knowledge of the TOE. This knowledge is usually public e.g. available by literature and/or the internet.

– "*Restricted*" means that the attacker possibly knows simple internal procedures of the TOE.

– In the case that an attacker knows significant relationships within the TOE and/or has special knowledge of parts of the TOE, the attacker could be seen as someone who has "*Sensitive*" information like detailed design knowledge (e.g. low-level design).

– This parameter is considered "*Critical*" if an attacker knows exactly how the TOE works internally (e.g. the source code) including details that can be used to exploit potential vulnerabilities.

596 Generally two possibilities exist to obtain knowledge of the TOE. Firstly an attacker has access to the development documentation or has knowledge of its content. In this case the question arises if the development environment could be deemed to be trustworthy or if it is imaginable that development information could leave the protected area. Secondly an attacker could become familiar with details of the TOE internals during the operational work.

597 In the second case, the time factor has to be considered. If the life expectancy (see definition above) of the TOE is shorter than the time an attacker (in consideration of his expertise) needs to get specific information of the TOE, the level of ***Knowledge of the TOE*** has to be set to a value which correlates with the respective knowledge the attacker can probably acquire during this available time.

598 The possibility to buy such kind of knowledge is not in the scope of this parameter. The parameter "Equipment" (discussed below) deals with this issue.

A.10.1.3 Knowledge of Exploitation

599 *The knowledge an attacker can obtain to develop and perform the exploitation of a potential vulnerability within the life expectancy of the TOE.*

600 On the basis of Knowledge of the Technology and ***Knowledge of the TOE*** an attacker can develop and perform attacks against the TOE or part of the

TOE to exploit potential vulnerabilities. Therefore, an attacker needs expertise of methods to organise and to carry out such attacks. Knowledge of Exploitation only deals with the knowledge of planning possible attacks and the capability to carry them out, but not with physically completing the attack (this is discussed under Opportunity below).

601 Similar to the definition of Knowledge of the Technology, four levels of knowledge (expertise) for exploitation of potential vulnerabilities are defined: *Inexperienced-Layman*, *Low-Experienced-Layman*, *Proficient Attacker* or *Expert*.

a) In this context, an *Inexperienced-Layman* is defined as someone who does not have any experience to develop, plan and/or perform attacks against the TOE whereas a *Low-Experienced-Layman* has little experience to do so, possibly assisted in similar attacks under instruction.

b) A *Proficient Attacker* has at minimum intermediate knowledge to exploit specific vulnerabilities of the TOE.

c) One can be sure to assume that an *Expert* has enough experience to design and perform (if the opportunity and equipment are available) exploitations of potential vulnerabilities.

602 The knowledge of an expert can be seen as something that represents the state of the art knowledge concerning a certain technology. With respect to specific exploitations of potential vulnerabilities of the TOE or parts of the TOE, this leads to the fact that if there are any vulnerabilities in the TOE an expert knows exactly what has to be done for their successfully exploitation.

603 Proceeding on the assumption that an attacker has the opportunity (see Opportunity) and the equipment (***Equipment***) to carry out an attack, Knowledge of Exploitation only depends on the capability and time required to organise this exploitation. The knowledge of the technology and of the TOE is discussed by the parameters Knowledge of the Technology and ***Knowledge of the TOE*** (see above). Therefore, an estimation of the expertise for developing, designing and performing exploitations has to be made under consideration of the life expectancy of the TOE. This means a specific attack must be practical within the available time by an attacker possessing the expertise estimated for Knowledge of Exploitation. In the case that the knowledge of an attacker is not sufficient to identify a potential vulnerability then the question arises whether the attacker is able to get this expertise within the life expectancy of the TOE. If this is possible a proficient attacker can become an expert attacker.

604 As already mentioned for ***Knowledge of the TOE*** the possibility to buy such kind of knowledge isn”t in the scope of this parameter. This subject is also addressed by parameter “***Equipment***”) discussed below.

A.10.1.4 Opportunity

605 *The likelihood of having the required access to the TOE within its life expectancy.*

606 Exploiting a potential vulnerability in the TOE requires some level of access to the TOE. In general the opportunity to have access to the TOE depends on the operational environment in which the TOE is used.

607 This parameter determines the following levels of access to the TOE: *easy, with some effort, difficult or improbable*. This parameter does not take into consideration whether an attacker has suitable equipment or the expertise to perform the exploitation of a potential vulnerability. These are discussed by ***Equipment*** and ***Knowledge of Exploitation respectively***.

a) If a TOE is generally available (e.g. available for members of the public to buy, without restrictions) then the opportunity to perform an attack is *easy* in principle.

b) In the case that a TOE is merely distributed or available for a limited circle but there are no high obstacles (e.g. there are no special protective measures restricting access to the TOE) to get one or more samples of it than the TOE is available with *some effort*.

c) If the operational environment of the TOE protects the TOE for example by organisational measures then it is *difficult* to get access to it.

d) It should be *improbable* to get the opportunity of access to the TOE if the TOE operates inside a high protected area with strict access rules, where only a few people that can be seen as very trustworthy have access to the TOE.

608 The determination for the level of opportunity in the context of this parameter reflects the probability that an unauthorised person can have access to the TOE within its life expectancy. Therefore, the time required for an attacker to exploit a vulnerability is the main factor on which ***Opportunity*** depends.

609 The nature of the TOE may also need to be considered when determining access to the TOE. For example, some TOEs may be rendered inoperable following an attempt to exploit a potential vulnerability. Therefore if multiple attempts are required, multiple copies of the TOE may also be required. The feasibility of obtaining multiple copies should be considered, as this may become cost prohibitive or there may be controls on attempts to obtain multiple copies.

A.10.1.5 Equipment

610 *The type of equipment necessary to exploit the potential vulnerability.*

611 To exploit a potential vulnerability of the TOE special equipment may be necessary. There are different types of equipment imaginable: Firstly the hardware and software devices needed to perform a certain attack against the TOE. Secondly the human resource required is another factor that could have an influence to drive a successful attack.

612 Money is an additional possibility to improve the prerequisite for that. Although the financial aspect could affect also the first types, it also could be used to buy other things (e.g. industrial espionage) which are helpful to perform attacks.

613 The following levels of equipment necessary to exploit a potential vulnerability are defined: *Standard*, *Higher-than-Average*, *Specialised* or *Bespoke*.

a) In this context the definition of "*Standard*" equipment is only the equipment (hardware, software, money or human resource) likely to be possessed by a standard user of the TOE that is necessary to perform an attack.

b) If some more effort is essential, e.g. special equipment (that means the equipment needed cannot be considered as a common object of utility by the TOE user) must be bought within a moderate price-range (one can finance this utility without major difficulty), than the value of ***Equipment*** can be set to "*Higher-than-Average*".

c) "*Specialised*" equipment does demand an expense and/or human resource that is difficult to realise by a private person.

d) In this context "*Bespoke*" means that the equipment and human resource needed is not commonly available.

614 Before an estimation of the equipment is determined, the interrelationships between the three types of equipment mentioned above must be clear because an assessment of ***Equipment*** must be done under consideration of all types of equipment required to realise an attack. For example if an attacker has enough money to buy all hardware and software components which are necessary but it seems to be impossible to get the human resource to carry out the attack then the level of equipment should be rated as "bespoke" due to the constraints in obtaining the human resource required.

A.10.2 Determination of the Attack Potential

615 On the basis of the independent parameter definitions it should be possible to decide whether a potential vulnerability is resistant against a certain attack potential.

616 To determine resistance, the user shall consider the TOE and the potential vulnerability with the definitions of the independent parameters (as described in Section A.10.1.

617 For a given potential vulnerability, the evaluator has to give a statement of the rating for each independent parameter. This rating is to be compared with the minimum rating of that parameter for a given attack potential. If for each parameter the rating assigned by the user meets or exceeds the minimum rating for the given attack potential the TOE is considered to resist the potential vulnerability.

618 To specify the attack potential the TOE is to resist, the PP/ST author identifies for each parameter the minimum level assumed to be possessed by an attacker of the TOE. The author will then compare these ratings to the minimums specified for each of the levels of attack potential to identify the appropriate attack potential claim.

619 A lookup table is provided, allowing a value to be identified for each independent parameter. For each parameter, the evaluator looks up the rating considered to be appropriate for the potential vulnerability in Table 13 Assignment of TOE's characteristic to the requirements derived from the definitions of the independent parameters .

Independent Parameter	Rating	Value
Knowledge of the Technology	Inexperienced-Layman	0
	Low-experience-Layman	1
	Proficient	2
	Expert	3
Knowledge of the TOE	None	0
	Restricted	1
	Sensitive	2
	Critical	3
Knowledge of Exploitation	Inexperienced-Layman	0
	Low-experience-Layman	1
	Proficient	2
	Expert	3
Opportunity	Easy	0
	Some Effort	1
	Difficult	2
	Improbable	3
Equipment	Standard	0
	Higher average	1
	Specialised	2
	Bespoke	3

Table 13 Assignment of TOE's characteristic to the requirements derived from the definitions of the independent parameters

620 Application steps using Table 13 Assignment of TOE's characteristic to the requirements derived from the definitions of the independent parameters :

a) Estimate for each independent parameter the requirements, as given in Section A.10.1, met by an attacker of the TOE.

b) For each parameter make the assignment to that rating which the TOE at least fulfilled. Determine the appropriate cells in column “Rating”.

621 There may be additional factors that prevent a vulnerability from being exploited, and therefore Table 13 Assignment of TOE's characteristic to the requirements derived from the definitions of the independent parameters does not need to be applied. For example, if a brute force attack against a permutational or probabilistic mechanism will take a longer to complete than the life-expectancy of the TOE, the evaluator reports that the vulnerability cannot be exploited in the TOE.

A.11 Determination of the Requirements for Attack Potential of a Potential Vulnerability

622 The following table contains on the one hand the minimum requirements of each independent parameter (represented by the rows) which on the other hand depends on the level of the attack potential of the identified vulnerability (represented by the columns).

623 To confirm that a vulnerability has a certain attack potential, one has to check whether the rating of each independent parameter of the analysed vulnerability fulfills the requirements necessary for a defined attack potential, i.e. for each parameter the assigned rating achieves at least the minimum stated in Table 14 Determination of the Attack Potential for the a given attack potential.

624 If the requirements, for a particular characteristic (expressed by an independent parameter) that is considered important to decide the attack potential rating, are not fulfilled the considered attack potential has not been achieved. This means that when applying this method it is not possible to compensate a “weak” estimated parameter by another parameter that was estimated as “High”. Each independent parameter is considered equally in a way that a minimum of requirements have to be met.

Independent Parameter	Attack Potential “Basic”	Attack Potential “Moderate”	Attack Potential “High”
Knowledge of the Technology	0	1	2
Knowledge of the TOE	1	2	3
Knowledge of Exploitation	1	2	3
Opportunity	1	2	3
Equipment	1	2	3

Table 14 Determination of the Attack Potential

625 Application steps using Table 14 Determination of the Attack Potential :

a) Take the assigned rating values resulted of step 2 described in Section A.10.2.

b) Make sure that for each parameter the minimum requirements displayed by the “Attack Potential” column which shall be claimed are fulfilled.

B Cross reference of assurance component dependencies

(informative)

626 The dependencies documented in the components of clauses 3-13 are the direct dependencies between the assurance components.

627 The following dependency tables for assurance components show their direct, indirect and optional dependencies. Each of the components that is a dependency of some assurance component is allocated a column. Each assurance component is allocated a row. The value in the table cell indicate whether the column label component is directly required (indicated by a cross "X") or indirectly required (indicated by a dash "-"), by the row label component. If no character is presented, the component is not dependent upon another component.

	ACM_CAP.	ALC_DVS.	ALC_DVS.
ACM_AUT.1	X	-	
ACM_AUT.2	X	-	
ACM_CAP.1			
ACM_CAP.2			
ACM_CAP.3		X	
ACM_CAP.4		X	
ACM_CAP.5			X
ACM_SCP.1	X	-	
ACM_SCP.2	X	-	
ACM_SCP.3	X	-	

Table 15 Dependency table for Class ACM: Configuration management

	ACM_CAP.	ADV_FSP.	ADV_RCR.	AGD_ADM	ALC_DVS.
ADO_DEL.1					
ADO_DEL.2	X				-
ADO_DEL.3	X				-
ADO_IGS.1		-	-	X	
ADO_IGS.2		-	-	X	

Table 16 Dependency table for Class ADO: Delivery and operation

	ADV_FSP.	ADV_FSP.	ADV_FSP.	ADV_HLD.	ADV_HLD.	ADV_HLD.	ADV_IMP.	ADV_IMP.	ADV_INT.	ADV_LLD.	ADV_RCR.	ADV_RCR.	ADV_RCR.	ALC_TAT.
ADV_FSP.1											X			
ADV_FSP.2											X			
ADV_FSP.3											X			
ADV_FSP.4											X			
ADV_HLD.1	X										X			
ADV_HLD.2	X										X			
ADV_HLD.3		X									-	X		
ADV_HLD.4		X									-	X		
ADV_HLD.5			X								-		X	
ADV_IMP.1	-			-			-			X	X			X
ADV_IMP.2	-			-			-			X	-			X
ADV_IMP.3	-			-			-		X	X	X			X
ADV_INT.1	-			-			X			X	-			-
ADV_INT.2	-			-			X			X	-			-
ADV_INT.3	-			-			-	X		X	-			-
ADV_LLD.1	-			X							X			
ADV_LLD.2		-			X						-	X		
ADV_LLD.3			-			X					-		X	
ADV_RCR.1														
ADV_RCR.2														
ADV_RCR.3														
ADV_SPM.1	X										-			
ADV_SPM.2	X										-			
ADV_SPM.3	X										-			

Table 17 Dependency table for Class ADV: Development

	ADV_FSP.	ADV_RCR.
AGD_ADM.1	X	-
AGD_USR.1	X	-

Table 18 Dependency table for Class AGD: Guidance documents

	ADV_FSP.	ADV_HLD.	ADV_IMP.	ADV_LLD.	ADV_RCR.	ALC_TAT.
ALC_DVS.1						
ALC_DVS.2						
ALC_FLR.1						
ALC_FLR.2						
ALC_FLR.3						
ALC_LCD.1						
ALC_LCD.2						
ALC_LCD.3						
ALC_TAT.1	-	-	X	-	-	-
ALC_TAT.2	-	-	X	-	-	-
ALC_TAT.3	-	-	X	-	-	-

Table 19 Dependency table for Class ALC: Life cycle support

	APE_ECD.	APE_INT.1	APE_OBJ.1	APE_SPD.1	ASE_ECD.	ASE_REQ.
APE_CCL.1		X			X	X
APE_ECD.1						
APE_INT.1						
APE_OBJ.1				X		
APE_REQ.1	X					
APE_REQ.2	X		X	-		
APE_SPD.1						

Table 20 Dependency table for Class APE: Protection Profile evaluation

	ASE_ECD.	ASE_INT.1	ASE_OBJ.1	ASE_REQ.	ASE_SPD.1
ASE_CCL.1	X	X	X	X	X
ASE_ECD.1					
ASE_INT.1					
ASE_OBJ.1					X
ASE_REQ.1	X				
ASE_REQ.2	X		X		-
ASE_SPD.1					
ASE_TSS.1	-	X		X	

Table 21 Dependency table for Class ASE: Security Target evaluation

	ADV_FSP.	ADV_HLD.	ADV_HLD.	ADV_IMP.	ADV_IMP.	ADV_LLD.	ADV_RCR.	AGD_ADM	AGD_USR.	ALC_TAT.	ATE_FUN.
ATE_COV.1	X						-				X
ATE_COV.2	X						-				X
ATE_COV.3	X						-				X
ATE_DPT.1	-	X					-				X
ATE_DPT.2	-		X			X	-				X
ATE_DPT.3	-		X	-	X	X	-			-	X
ATE_FUN.1											
ATE_FUN.2											
ATE_IND.1	X						-	X	X		
ATE_IND.2	X						-	X	X		X
ATE_IND.3	X						-	X	X		X

Table 22 Dependency table for Class ATE: Tests

	ADO_IGS.1	ADV_FSP.	ADV_FSP.	ADV_HLD.	ADV_IMP.	ADV_IMP.	ADV_LLD.	ADV_RCR.	AGD_ADM	AGD_USR.	ALC_TAT.
AVA_CCA.1		-	X	-	-	X	-	-	X	X	-
AVA_CCA.2		-	X	-	-	X	-	-	X	X	-
AVA_CCA.3		-	X	-	-	X	-	-	X	X	-
AVA_MSU.1	X	X						-	X	X	
AVA_MSU.2	X	X						-	X	X	
AVA_MSU.3	X	X						-	X	X	
AVA_VLA.1											
AVA_VLA.2											
AVA_VLA.3											
AVA_VLA.4											

Table 23 Dependency table for Class AVA: Vulnerability assessment

C Cross reference of EALs and assurance components

(informative)

628 Table 24 Evaluation assurance level summary describes the relationship between the evaluation assurance levels and the assurance classes, families and components.

Assurance class	Assurance Family	Assurance Components by Evaluation Assurance Level						
		EAL1	EAL2	EAL3	EAL4	EAL5	EAL6	EAL7
Configuration management	ACM_AUT				**1**	1	**2**	2
	ACM_CAP	**1**	**2**	**3**	**4**	4	**5**	5
	ACM_SCP			**1**	**2**	**3**	3	3
Delivery and operation	ADO_DEL		**1**	1	**2**	2	2	**3**
	ADO_IGS	**1**	1	1	1	1	1	1
Development	ADV_FSP	**1**	1	1	**2**	**3**	3	**4**
	ADV_HLD		**1**	**2**	2	**3**	**4**	**5**
	ADV_IMP				**1**	**2**	**3**	3
	ADV_INT					**1**	**2**	**3**
	ADV_LLD				**1**	1	**2**	2
	ADV_RCR	**1**	1	1	1	**2**	2	**3**
	ADV_SPM				**1**	**3**	3	3
Guidance documents	AGD_ADM	**1**	1	1	1	1	1	1
	AGD_USR	**1**	1	1	1	1	1	1
Life cycle support	ALC_DVS			**1**	1	1	**2**	2
	ALC_FLR							
	ALC_LCD				**1**	**2**	2	**3**
	ALC_TAT				**1**	**2**	**3**	3
Security Target evaluation	ASE_CCL	**1**	1	1	1	1	1	1
	ASE_ECD	**1**	1	1	1	1	1	1
	ASE_INT	**1**	1	1	1	1	1	1
	ASE_OBJ		**1**	1	1	1	1	1
	ASE_REQ	**1**	**2**	2	2	2	2	2
	ASE_SPD		**1**	1	1	1	1	1
	ASE_TSS	**1**	1	1	1	1	1	1
Tests	ATE_COV		**1**	**2**	2	2	**3**	3
	ATE_DPT			**1**	1	**2**	2	**3**
	ATE_FUN		**1**	1	1	1	**2**	2
	ATE_IND	**1**	**2**	2	2	2	2	**3**
Vulnerability assessment	AVA_CCA					**1**	**2**	2
	AVA_MSU			**1**	**2**	2	**3**	3
	AVA_VLA		**1**	1	**2**	**3**	**4**	4

Table 24 Evaluation assurance level summary