Australian Government
Department of Defence

Defence Signals Directorate

Australasian Information Security Evaluation Program

Depth (ATE_DPT.3) - CC V2.2

Common Evaluation Methodology

21 December 2005

Version 1.1

Amendment Record

Version	Date	Description
1.0	12 December 2005	Released.
1.1	21 December 2005	Releasable to AISEFs.

Table of Contents

1 Testing: implementation representation (ATE_DPT.3)

1.1 Objectives

1 The subsystems of a TSF provide a high-level description of the internal workings of the TSF. Testing at the level of the subsystems, in order to demonstrate the presence of any flaws, provides assurance that the TSF subsystems have been correctly realised.

2 The modules of a TSF provide a description of the internal workings of the TSF. Testing at the level of the modules, in order to demonstrate the presence of any flaws, provides assurance that the TSF modules have been correctly realised.

3 The implementation representation of a TSF provides a detailed description of the internal workings of the TSF. Testing at the level of the implementation, in order to demonstrate the presence of any flaws, provides assurance that the TSF implementation has been correctly realised.

4 The objective of this sub-activity is to determine whether the developer has tested to the depth of the implementation representation.

1.2 Input

5 The evaluation evidence for this sub-activity is:

a) the ST;

b) the functional specification;

c) the high-level design;

d) the low-level design;

e) the implementation representation;

f) the test documentation;

g) the depth of testing analysis.

1.3 Evaluator Actions

1.3.1 ATE_DPT.3.1E

ATE_DPT.3.1C	The depth analysis shall demonstrate that the tests identified in the test documentation are sufficient to demonstrate that the TSF operates in accordance with its high-level design, low-level design **and implementation representation**.

ATE_DPT.3-1 The evaluator ***shall examine*** the depth of testing analysis for a mapping between the tests identified in the test documentation and the high-level design and the low-level design.

6 The depth of testing analysis identifies all subsystems described in the high-level design and all modules of the low-level design and provides a mapping of the tests to these subsystems and modules. Correspondence may take the form of a table or matrix. In some cases the mapping may be sufficient to show test correspondence. In other cases a rationale (typically prose) may have to supplement the mapping evidence provided by the developer.

7 All design details specified in the high-level design and low-level design that map to and satisfy TOE security requirements are subject to testing and hence, should be mapped to test documentation. Figure 1 displays a conceptual framework (excluding the implementation representation) of the mapping between subsystems described in the high-level design and modules of the low-level design and the tests outlined in the test documentation used to test them. Tests may involve one or multiple security functions depending on the test dependencies or the overall goal of the test being performed.

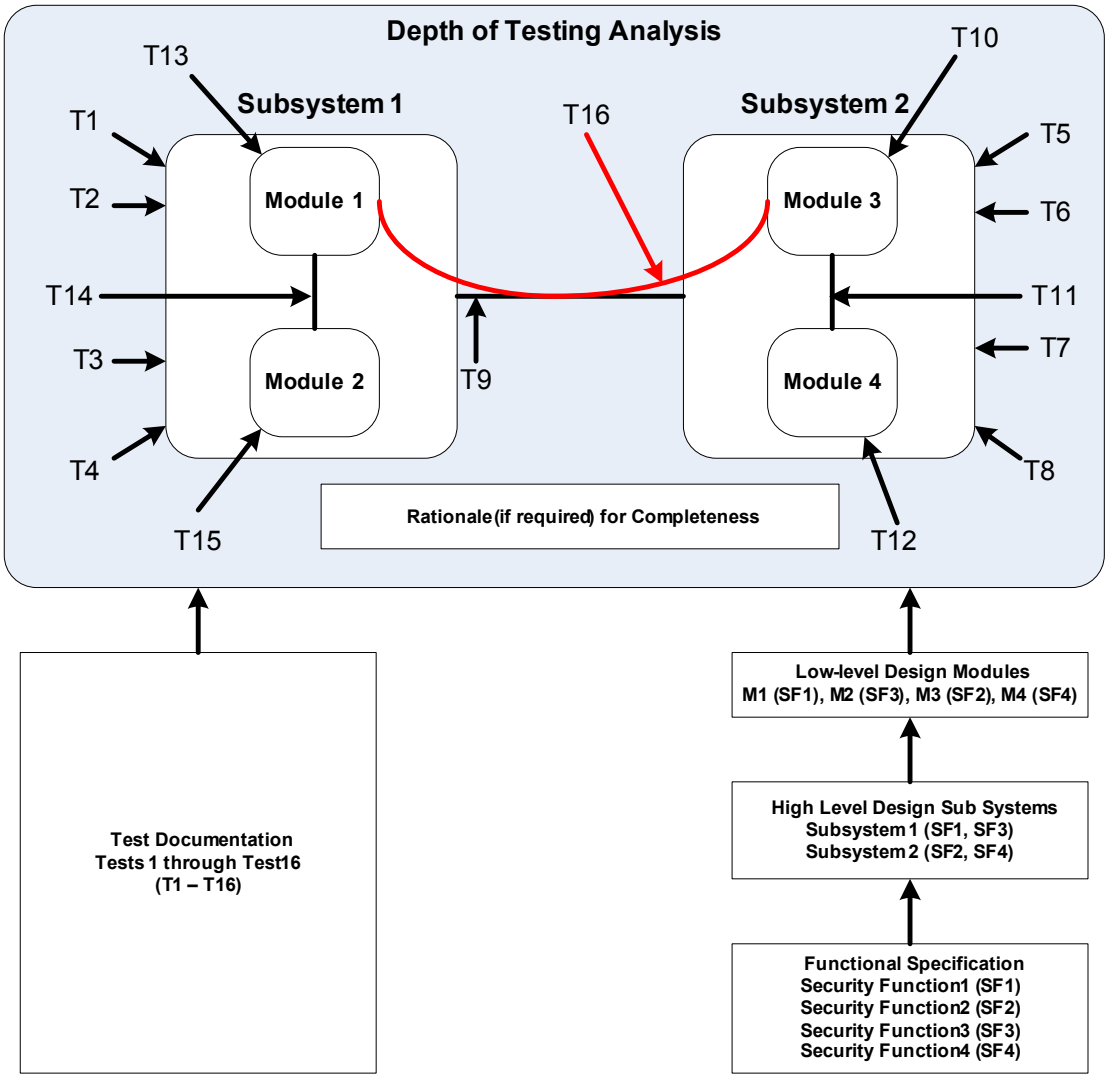

Figure 1: Depth of testing conceptual framework

ATE_DPT.3-2 The evaluator ***shall examine*** the depth of testing analysis for a mapping between the tests identified in the test documentation and the TSF implementation representation.

8 Portions of the implementation representation (such as source code methods, functions and procedures or groupings of hardware components) that implement the TSF should be subject to testing. At this level, testing of portions of the TSF implementation representation is analogous to unit testing before incorporation into modules and sub-systems of the TSF. The depth of testing analysis must map tests identified in the test documentation to those portions of the TSF implementation representation that are the subject of testing.

9 For the conceptual framework illustrated in Figure 1 depth of testing requires that those portions of the implementation representation that implement modules of the TSF be tested. The implementation representation has not been included in the figure to reduce complexity of the diagram.

ATE_DPT.3-3 The evaluator ***shall examine*** the test documentation to determine that the testing approach for each security function of the TSF is suitable to demonstrate the expected behaviour.

10 Before the adequacy of test documentation can be accurately evaluated, or before new tests can be created, the evaluator has to understand the desired expected behaviour of a security function in the context of the requirements it is to satisfy.

11 The evaluator may choose to focus on one security function of the TSF at a time. For each security function, the evaluator reviews the ST requirement and the relevant parts of the functional specification, high-level design, low-level design, implementation representation and guidance documentation to gain an understanding of the way the TOE is expected to behave.

12 With an understanding of the expected behaviour, the evaluator reviews the test documentation to gain an understanding of the testing approach. In most cases, the testing approach will entail a security function being stimulated at either the external or internal interfaces and its responses are observed. However, there may be cases where a security function cannot be adequately tested at an interface (as may be the case, for instance, for residual information protection functionality); in such cases, other means will need to be employed.

13 In cases where it is impractical or inadequate to test at an interface, the test documentation should identify the alternate approach to verify expected behaviour. It is the evaluator's responsibility to determine the suitability of the alternate approach. However, the following should be considered when assessing the suitability of alternate approaches:

a) an analysis of the implementation representation to determine that the required behaviour should be exhibited by the TOE is an acceptable alternate approach. This could mean a code inspection for a software TOE or perhaps a chip mask inspection for a hardware TOE.

b) it is acceptable to use evidence of developer integration or module testing, even if the EAL is not commensurate with evaluation exposure to the low-level design or implementation. If evidence of developer integration or module testing is used in verifying the expected behaviour of a security function, care should be given to confirm that the testing evidence reflects the current implementation of the TOE. If the subsystem or modules have been changed since testing occurred, evidence that the changes were tracked and addressed by analysis or further testing will usually be required.

14 It should be emphasised that supplementing the testing effort with alternate approaches should only be undertaken when both the developer and evaluator determine that there exists no other practical means to test the expected behaviour of a security function. This alternative is made available to the developer to minimize the cost (time and/or money) of testing under the circumstances described above; it is not designed to give the evaluator more latitude to demand unwarranted additional information about the TOE, nor to replace testing in general.

15 Testing of the TSF may be performed at the external interfaces, internal interfaces, or a combination of both. Whatever strategy is used the evaluator will consider its appropriateness for adequately testing the security functions. Specifically the evaluator determines whether testing at the internal interfaces for a security function is necessary or whether these internal interfaces can be adequately tested (albeit implicitly) by exercising the external interfaces. This determination is left to the evaluator, as is its justification.

ATE_DPT.3-4 The evaluator ***shall examine*** the test procedures to determine that the test prerequisites, test steps and expected result(s) adequately test each security function.

16 Test pre-requisites are necessary to establish the required initial conditions for the test. They may be expressed in terms of parameters that must be set or in terms of test ordering in cases where the completion of one test establishes the necessary pre-requisites for another test. The evaluator must determine that the pre-requisites are complete and appropriate in that they will not bias the observed test results towards the expected test results.

17 The test steps and expected results specify the actions and parameters to be applied to the interfaces as well as how the expected results should be verified and what they are. The evaluator must determine that the test steps and expected results are consistent with the functional specification, the high-level design, the low-level design and the implementation representation. The tests must verify behaviour documented in these specifications. This means that each security functional behaviour characteristic explicitly described in the functional specification, high-level design, low-level design and implementation representation should have tests and expected results to verify that behaviour.

18 Although all of the TSF has to be tested by the developer, exhaustive specification testing of the interfaces will be determined during the ATE_COV.X sub-activity. The overall aim of this activity is to determine that each security function has been sufficiently tested against the behavioural claims in the functional specification, the high-level design, the low-level design and implementation representation. The test procedures will provide insight as to how the security functions have been exercised by the developer during testing. The evaluator will use this information when developing additional tests to independently test the TOE.

ATE_DPT.3-5 The evaluator ***shall check*** the depth of testing analysis to ensure that the TSF as defined in the high-level design, low-level design and implementation representation is completely mapped to the tests in the test documentation.

19 The depth of testing analysis provides a complete statement of correspondence between the high-level design, the low-level design, the implementation representation and the test plan and procedures. The depth of testing analysis must consider:

a) All subsystems and internal interfaces described in the high-level design;

b) All modules and internal interfaces described in the low-level design; and

c) All portions of the TSF implementation representation.

20 All the subsystems, modules, internal interfaces and all portions of the TSF implementation representation are considered in the depth of testing analysis and must have tests mapped to them in order for completeness to be claimed. It would be expected that tests mapped to portions of the TSF implementation representation should correspond to modules or discrete items of functionality within modules of the TSF. The mapping of tests to modules or discrete items of functionality within modules for the TSF implementation representation would largely depend on the complexity of the TSF design, the nature of the TOE (i.e. hardware, software or a combination of both) and the developer's testing approach.

21 Incomplete coverage would be evident if a subsystem, module, internal interface or a portion of the TSF implementation representation was identified in the depth of testing analysis and no tests could be attributed to it.