Australian Government
Department of Defence

Defence Signals Directorate

Australasian Information Security Evaluation Program

Functional Tests (ATE_FUN.2) - 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 Ordered Functional Testing (ATE_FUN.2)

1.1 Objectives

1 The objective of this sub-activity it to determine whether the developer has demonstrated that all security functions perform as specified. The developer is required to perform testing and to provide test documentation.

2 An additional objective is to ensure that testing is structured such as to avoid circular arguments about the correctness of the portions of the TSF being tested.

1.2 Application Notes

3 The extent to which the test documentation is required to cover the TSF is dependent upon the coverage assurance component.

4 For the developer tests provided, the evaluator determines whether the tests are repeatable, and the extent to which the developer's tests can be used for the evaluator's independent testing effort. Any security function for which the developer's test results indicate that it may not perform as specified should be tested independently by the evaluator to determine whether or not it does.

1.3 Input

5 The evaluation evidence for this sub-activity is:

a) the ST;

b) the functional specification;

c) the test documentation.

1.4 Evaluator Actions

1.4.1 ATE_FUN.2.1E

ATE_FUN.2.1C	The test documentation shall consist of test plans, test procedure descriptions, expected test results and actual test results.

ATE_FUN.2-1 The evaluator ***shall check*** that the test documentation includes test plans, test procedure descriptions, expected test results and actual test results.

6 The evaluator reviews the test documentation to ensure that sufficient test documentation has been provided.

ATE_FUN.2.2C	The test plans shall identify the security functions to be tested and describe the goal of the tests to be performed.

ATE_FUN.2-2 The evaluator ***shall check*** that the test plan identifies the security functions to be tested.

7 One method that could be used to identify the security function to be tested is a reference to the appropriate part(s) of the functional specification that specifies the particular security function.

8 The evaluator may wish to employ a sampling strategy when performing this work unit.

9 For guidance on sampling see B.2.

ATE_FUN.2-3 The evaluator ***shall examine*** the test plan to determine that it describes the goal of the tests performed.

10 The test plan provides information about how the security functions are tested and the test configuration in which testing occurs.

11 The evaluator may wish to employ a sampling strategy when performing this work unit.

12 For guidance on sampling see B.2.

ATE_FUN.2-4 The evaluator ***shall examine*** the test plan to determine that the TOE test configuration is consistent with the configuration identified for evaluation in the ST.

13 The TOE referred to in the developer's test plan should have the same unique reference as established by the CM capabilities (ACM_CAP).* sub-activity.

14 It is possible for the ST to specify more than one configuration for evaluation. The TOE may be composed of a number of distinct hardware and software implementations that need to be tested in accordance with the ST. The evaluator verifies that there are test configurations identified in the developer test documentation that ate consistent with each evaluated configuration described in the ST.

15 The evaluator should consider the assumptions about the security aspects of the TOE environment described in the ST that may apply to the test environment. There may be some assumptions in the ST that do not apply to the test environment. For example, an assumption about user clearances may not apply; however, an assumption about a single point of connection to a network would apply.

ATE_FUN.2.3C The test procedure descriptions shall identify the tests to be performed and describe the scenarios for testing each security function. These scenarios shall include any ordering dependencies on the results of other tests.

ATE_FUN.2-5 The evaluator ***shall examine*** the test plan to determine that it is consistent with the test procedure descriptions.

16 The evaluator may wish to employ a sampling strategy when performing this work unit.

17 For guidance on sampling see B.2. For guidance on consistency analysis see B.3.

ATE_FUN.2-6 The evaluator ***shall check*** that the test procedure descriptions identify each security function behaviour to be tested.

18 One method that may be used to identify the security function behaviour to be tested is a reference to the appropriate part(s) of the design specification that specifies the particular behaviour to be tested.

19 The evaluator may wish to employ a sampling strategy when performing this work unit.

20 For guidance on sampling see B.2.

ATE_FUN.2-7 The evaluator ***shall examine*** the test procedure descriptions to determine that sufficient instructions are provided to establish reproducible initial test conditions including ordering dependencies if any.

21 Some steps may have to be performed to establish initial conditions. For example, user accounts need to be added before they can be deleted. An example of ordering dependencies on the results of other tests is the need to test the audit function before relying on it to produce audit records for another security mechanism such as access control. Another example of an ordering dependency would be where one test case generates a file of data to be used as input for another test case.

22 The evaluator may wish to employ a sampling strategy when performing this work unit.

23 For guidance on sampling see B.2.

ATE_FUN.2-8 The evaluator ***shall examine*** the test procedure descriptions to determine that sufficient instructions are provided to have a reproducible means to stimulate the security functions and to observe their behaviour.

24 Stimulus is usually provided to a security function externally through the TSFI. Once an input (stimulus) is provided to the TSFI, the behaviour of the security function can then be observed at the TSFI. Reproducibility is not assured unless the test procedures contain enough detail to unambiguously describe the stimulus and the behaviour expected as a result of this stimulus.

25 The evaluator may wish to employ a sampling strategy when performing this work unit.

26 For guidance on sampling see B.2.

ATE_FUN.2-9 The evaluator ***shall examine*** the test procedure descriptions to determine that they are consistent with the test procedures.

27 If the test procedure descriptions are the test procedures, then this work unit is not applicable and is therefore considered to be satisfied.

28 The evaluator may wish to employ a sampling strategy when performing this work unit.

29 For guidance on sampling see B.2. For guidance on consistency analysis see B.3.

ATE_FUN.2.4C The expected test results shall show the anticipated outputs from a successful execution of the tests.

ATE_FUN.2-10 The evaluator ***shall examine*** the test documentation to determine that sufficient expected tests results are included.

30 The expected test results are needed to determine whether or not a test has been successfully performed. Expected test results are sufficient if they are unambiguous and consistent with expected behaviour given the testing approach.

31 The evaluator may wish to employ a sampling strategy when performing this work unit.

32 For guidance on sampling see B.2.

ATE_FUN.2.5C The test results from the developer execution of the tests shall demonstrate that each tested security function behaved as specified.

ATE_FUN.2-11 The evaluator ***shall check*** that the expected test results in the test documentation are consistent with the actual test results provided.

33 A comparison of the actual and expected test results provided by the developer will reveal any inconsistencies between the results.

34 It may be that a direct comparison of actual results cannot be made until some data reduction or synthesis has been first performed. In such cases, the developer's test documentation should describe the process to reduce or synthesize the actual data.

35 For example, the developer may need to test the contents of a message buffer after a network connection has occurred to determine the contents of the buffer. The message buffer will contain a binary number. This binary number would have to be converted to another form of data representation in order to make the test more meaningful. The conversion of this binary representation of data into a higher-level representation will have to be described by the developer in enough detail to allow an evaluator to perform the conversion process (i.e. synchronous or asynchronous transmission, number of stop bits, parity, etc.).

36 It should be noted that the description of the process used to reduce or synthesize the actual data is used by the evaluator not to actually perform the necessary modification but to assess whether this process is correct. It is up to the developer to transform the expected test results into a format that allows an easy comparison with the actual test results.

37 The evaluator may wish to employ a sampling strategy when performing this work unit.

38 For guidance on sampling see B.2.

39 If the expected and actual test results for any test are not the same, then a demonstration of the correct operation of a security function has not been achieved. Such an occurrence will influence the evaluator's independent testing effort to include testing the implicated security function. The evaluator should also consider increasing the sample of evidence upon which this work unit is performed.

ATE_FUN.2-12 The evaluator ***shall report*** the developer testing effort, outlining the testing approach, configuration, depth and results.

40 The developer testing information recorded in the ETR allows the evaluator to convey the overall testing approach and effort expended on the testing of the TOE by the developer. The intent of providing this information is to give a meaningful overview of the developer testing effort. It is not intended that the information regarding developer testing in the ETR be an exact reproduction of specific test steps or results of individual tests. The intention is to provide enough detail to allow other evaluators and overseers to gain some insight about the developer's testing approach, amount of testing performed, TOE test configurations, and the overall results of the developer testing.

41 Information that would typically be found in the ETR section regarding the developer testing effort is:

a) TOE test configurations. The particular configurations of the TOE that were tested;

b) testing approach. An account of the overall developer testing strategy employed;

c) amount of developer testing performed. A description on the extent of coverage and depth of developer testing;

d) testing results. A description of the overall developer testing results.

42 This list is by no means exhaustive and is only intended to provide some context as to the type of information that should be present in the ETR concerning the developer testing effort.

ATE_FUN.2.6C The test documentation shall include an analysis of the test procedure ordering dependencies.

ATE_FUN.2-13 The evaluator ***shall check*** that the test documentation includes an analysis of the test procedure ordering dependencies.

43 Although the test procedures may state pre-requisite initial test conditions in terms of ordering of tests, they may not have provided a rationale for the ordering.

44 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.

45 For this sub-activity, the developer's test documentation must include such an analysis.

ATE_FUN.2-14 The evaluator ***shall examine*** the analysis of the test procedure ordering dependencies to determine that the analysis is complete.

46 The evaluator will have examined ordering dependencies in the ATE_FUN.2-7. All ordering dependencies that are identified in the test procedures must be considered in the developer's dependency analysis.

47 Where there are no dependencies, the analysis shall explain why there is no requirement for dependencies.

ATE_FUN.2-15 The evaluator ***shall examine*** the analysis of the test procedure ordering dependencies to determine that it contains a rationale for why it is appropriate for the tests to be ordered as they are.

48 The evaluator reviews the developer's rationale for test procedure ordering dependencies looking for errors or omissions that may result in concealment of a fault in the implementation of the TSF.

49 As an example, one test case may generate a file of data to be used as input for another subsequent test case. However, this generated file may not contain the appropriate input parameters to exercise all security behaviours observable in the subsequent test case. In this circumstance, a fault may be present in that portion of the implementation not exercised and therefore concealed by the testing. This could be considered a fault in the ordering of test cases.

50 Another example would be where a test case places the TSF into a state that causes a subsequent test case to be adversely affected. In this case, a fault in the implementation may be concealed.