Australian Government
Department of Defence

Defence Signals Directorate

Australasian Information Security Evaluation Program

Highly Resistant (AVA_VLA.4) - CC V2.2

Common Evaluation Methodology

10 February 2006

Version 1.1

Amendment Record

Version	Date	Description
1.0	1 February 2006	Released.
1.1	10 February 2006	Releasable to AISEFs.

Table of Contents

1 Highly Resistant

1.1 Objectives

1 A vulnerability analysis is performed by the developer to ascertain the presence of security vulnerabilities, and to confirm that they cannot be exploited in the intended environment for the TOE.

2 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 high attack potential.

1.2 Input

3 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 subset of the implementation representation;

f) the architectural description;

g) the TOE security policy model;

h) the user guidance;

i) the administrator guidance;

j) the secure installation, generation, and start-up procedures;

k) the vulnerability analysis;

l) the strength of function claims analysis;

m) the covert channel analysis;

n) tests results of functional tests.

o) the TOE suitable for testing.

4 Other input for this sub-activity is:

a) current information regarding public domain vulnerabilities and attacks.

1.3 Evaluator Actions

1.3.1 AVA_VLA.4.1E

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 vulnerabilities.
AVA_VLA.4.3C	The vulnerability analysis documentation shall show, for all identified vulnerabilities, that the vulnerability cannot be exploited in the intended environment for the TOE.
AVA_VLA.4.4C	The vulnerability analysis documentation shall justify that the TOE, with the identified vulnerabilities, is resistant to obvious penetration attacks.

AVA_VLA.4-1 The evaluator ***shall examine*** the developer's vulnerability analysis to determine that the search for vulnerabilities has considered all relevant information.

5 The developer's vulnerability analysis should cover the developer's search for vulnerabilities in at least all evaluation deliverables and public domain information sources.

6 Information in the public domain is highly dynamic. Therefore, it is possible that new vulnerabilities are reported in the public domain between the time the developer performs the vulnerability analysis and the time that the evaluation is completed. The point at which monitoring of the public domain information ceases is an evaluation authority issue; therefore guidance and agreement should be sought from the evaluation authority.

AVA_VLA.4-2 The evaluator ***shall examine*** the developer's vulnerability analysis to determine that each identified vulnerability is described and that a rationale is given for why it is not exploitable in the intended environment for the TOE.

7 A vulnerability is termed non-exploitable if one or more of the following conditions exist:

a) security functions or measures in the (IT or non-IT) environment prevent exploitation of the vulnerability in the intended environment. For instance, restricting physical access to the TOE to authorised users only may effectively render a TOE's vulnerability to tampering unexploitable;

b) the vulnerability is exploitable, however execution of the exploit is infeasible, due to impractical requirements of time or other factor. A rationale shall justify why an attack is impractical. Such vulnerabilities are reported in the ETR as residual vulnerabilities;

c) either the threat is not claimed to be countered or the violable organisational security policy is not claimed to be achieved by the ST. For instance, a firewall whose ST makes no availability policy claim and is vulnerable to TCP SYN attacks (an attack on a common Internet protocol that renders hosts incapable of servicing connection requests) should not fail this evaluator action on the basis of this vulnerability alone.

8 For guidance on determining attack potential necessary to exploit a vulnerability see B.8.

AVA_VLA.4-3 The evaluator ***shall examine*** the developer's vulnerability analysis to determine that it is consistent with the ST and the guidance.

9 The developer's vulnerability analysis may address a vulnerability by suggesting specific configurations or settings for TOE functions. If such operating constraints are deemed to be effective and consistent with the ST, then all such configurations/settings should be adequately described in the guidance so that they may be employed by the consumer.

AVA_VLA.4.5C The vulnerability analysis documentation shall show that the search for vulnerabilities is systematic.

AVA_VLA.4-4 The evaluator ***shall examine*** the developer's vulnerability analysis to determine that it is systematic.

10 A vulnerability analysis is considered to be systematic, if it exhibits the following characteristics:

a) it follows a predetermined, planned approach;

b) it is repeatable;

c) it employs a method whereby completeness of the items being relevant for the analysis can be determined.

d) it identifies of all TOE documentation upon which the search for flaws was based.

11 An approach based on brainstorming techniques would not meet this requirement unless the results were analysed to ensure that all areas of potential vulnerability had been considered.

12 The associated evidence that the search for vulnerabilities was systematic should include identification of all TOE documentation upon which the search for flaws was based.

AVA_VLA.4.6C The vulnerability analysis documentation shall provide a justification that the analysis completely addresses the TOE deliverables.

AVA_VLA.4-5 The evaluator ***shall examine*** the developer's vulnerability analysis to determine that it provides a justification that the analysis completely addresses the TOE Deliverables.

13 The developer's vulnerability analysis is required to consider all TOE deliverables submitted for evaluation. The developer needs to justify that each TOE deliverable has been analysed for potential vulnerabilities.

1.3.2 AVA_VLA.4.2E

AVA_VLA.4.2E The evaluator shall conduct penetration testing, building on the developer vulnerability analysis, to ensure the identified vulnerabilities have been addressed.

AVA_VLA.4-6 The evaluator ***shall devise*** penetration tests, building on the developer vulnerability analysis.

14 The evaluator prepares for penetration testing:

a) as necessary to attempt to disprove the developer's analysis in cases where the developer's rationale for why a vulnerability is unexploitable is suspect in the opinion of the evaluator;

b) as necessary to determine the susceptibility of the TOE, in its intended environment, to a vulnerability not considered by the developer. The evaluator should have access to current information (e.g. from the overseer) regarding public domain vulnerabilities that may not have been considered by the developer, and may also have identified potential vulnerabilities as a result of performing other evaluation activities.

15 With an understanding of the suspected vulnerability, the evaluator determines the most feasible way to test for the TOE's susceptibility. Specifically the evaluator considers:

a) the security function interfaces that will be used to stimulate the TSF and observe responses;

b) initial conditions that will need to exist for the test (i.e. any particular objects or subjects that will need to exist and security attributes they will need to have);

c) special test equipment that will be required to either stimulate a security function or make observations of a security function.

16 The evaluator will probably find it practical to carry out penetration testing using a series of test cases, where each test case will test for a specific vulnerability.

AVA_VLA.4-7 The evaluator ***shall produce*** penetration test documentation for the tests that build upon the developer vulnerability analysis, in sufficient detail to enable the tests to be repeatable. The test documentation shall include:

a) identification of the vulnerability the TOE is being tested for;

b) instructions to connect and setup all required test equipment as required to conduct the penetration test;

c) instructions to establish all penetration test prerequisite initial conditions;

d) instructions to stimulate the TSF;

e) instructions for observing the behaviour of the TSF;

f) descriptions of all expected results and the necessary analysis to be performed on the observed behaviour for comparison against expected results;

g) instructions to conclude the test and establish the necessary post-test state for the TOE.

17 The intent of specifying this level of detail in the test documentation is to allow another evaluator to repeat the tests and obtain an equivalent result.

AVA_VLA.4-8 The evaluator ***shall conduct*** penetration testing, building on the developer vulnerability analysis.

18 The evaluator uses the penetration test documentation resulting from work unit AVA_VLA.4-6 as a basis for executing penetration tests on the TOE, but this does not preclude the evaluator from performing additional ad hoc penetration tests. If required, the evaluator may devise ad hoc tests as a result of information learned during penetration testing that, if performed by the evaluator, are to be recorded in the penetration test documentation. Such tests may be required to follow up unexpected results or observations, or to investigate potential vulnerabilities suggested to the evaluator during the preplanned testing.

AVA_VLA.4-9 The evaluator ***shall record*** the actual results of the penetration tests.

19 While some specific details of the actual test results may be different from those expected (e.g. time and date fields in an audit record) the overall result should be identical. Any differences should be justified.

AVA_VLA.4-10 The evaluator ***shall report*** in the ETR the evaluator penetration testing efforts, outlining the testing approach, configuration, depth and results.

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

21 Information that would typically be found in the ETR section regarding evaluator penetration testing efforts is:

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

b) security functions penetration tested. A brief listing of the security functions that were the focus of the penetration testing;

c) verdict for the sub-activity. The overall judgement on the results of penetration testing.

22 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 penetration testing the evaluator performed during the evaluation.

1.3.3 AVA_VLA.4.3E

AVA_VLA.4.3E	The evaluator shall perform an independent vulnerability analysis.

AVA_VLA.4-11 The evaluator ***shall examine*** all inputs to this sub-activity to determine possible security vulnerabilities not already addressed by the developer's vulnerability analysis.

23 A flaw hypothesis methodology should be used whereby specifications and documentation for the TOE are analysed and then vulnerabilities in the TOE are hypothesised, or speculated. The list of hypothesised vulnerabilities is then prioritised on the basis of the estimated probability that a vulnerability exists and, assuming a vulnerability does exist, the attack potential required to exploit it, and on the extent of control or compromise it would provide. The prioritised list of potential vulnerabilities is used to direct penetration testing against the TOE.

24 For guidance on determining attack potential necessary to exploit a vulnerability see Annex B.8 of CEM part 2.

25 Vulnerabilities hypothesised exploitable by an attacker possessing a high attack potential, that do not result in a violation of the security objectives specified in the ST, do not result in a failure of this evaluator action. Where analysis supports the hypothesis, these need not be considered further as an input to penetration testing.

26 Vulnerabilities hypothesised as potentially exploitable by an attacker possessing a low or moderate or high attack potential and resulting in a violation of the security objectives should be the highest priority potential vulnerabilities comprising the list used to direct penetration testing against the TOE.

27 Subject to the threats being present in the intended environment, the evaluator's independent vulnerability analysis should consider generic vulnerabilities under each of the following headings:

a) generic vulnerabilities relevant for the type of TOE being evaluated, as may be supplied by the overseer;

b) bypassing;

c) tampering;

d) direct attacks;

e) misuse.

28 Items b) - e) are now explained in greater detail.

Bypassing

29 Bypassing includes any means by which an attacker could avoid security enforcement, by:

a) exploiting the capabilities of interfaces to the TOE, or of utilities which can interact with the TOE;

b) inheriting privileges or other capabilities that should otherwise be denied;

c) (where confidentiality is a concern) reading sensitive data stored or copied to inadequately protected areas.

30 Each of the following should be considered (where relevant) in the evaluator's independent vulnerability analysis.

a) Attacks based on exploiting the capabilities of interfaces or utilities generally take advantage of the absence of the required security enforcement on those interfaces. For example, gaining access to functionality that is implemented at a lower level than that at which access control is enforced. Relevant items include:

i) changing the predefined sequence of invocation of functions;

ii) executing an additional function;

iii) using a component in an unexpected context or for an unexpected purpose;

iv) using implementation detail introduced in less abstract representations;

v) using the delay between time of access check and time of use.

b) Changing the predefined sequence of invocation of components should be considered where there is an expected order in which interfaces to the TOE (e.g. user commands) are called to perform some security function (e.g. opening a file for access and then reading data from it). If a security function is invoked on one of the TOE interfaces (e.g. an access control check), the evaluator should consider whether it is possible to bypass the security function by performing the call at a later point in the sequence or by missing it out altogether.

c) Executing an additional component (in the predefined sequence) is a similar form of attack to the one just described, but involves the calling of some other TOE interface at some point in the sequence. It can also involve attacks based on interception of sensitive data passed over a network by use of network traffic analysers (the additional component here being the network traffic analyser).

d) Using a component in an unexpected context or for an unexpected purpose includes using an unrelated TOE interface to bypass a security function by using it to achieve a purpose that it was not designed or intended to achieve. Covert channels are an example of this type of attack. The use of undocumented interfaces (which may be insecure) also falls into this category (these include undocumented support and help facilities).

e) Using implementation detail introduced in lower representations again includes the use of covert channels in which an attacker takes advantage of additional functions, resources or attributes that are introduced to the TOE as a consequence of the refinement process (e.g. use of a lock variable as a covert channel). Additional functionality may also include test harness code contained in software modules.

f) Using the delay between time of check and time of use includes scenarios where an access control check is made and access granted, and an attacker is subsequently able to create conditions in which, had they applied at the time the access check was made, would have caused the check to fail. An example would be a user creating a background process to read and send highly sensitive data to the user's terminal, and then logging out and logging back in again at a lower sensitivity level. If the background process is not terminated when the user logs off, the MAC checks would have been effectively bypassed.

g) Attacks based on inheriting privileges are generally based on illicitly acquiring the privileges or capabilities of some privileged component, usually by exiting from it in an uncontrolled or unexpected manner. Relevant items include:

   i) executing data not intended to be executable, or making it executable;

   ii) generating unexpected input for a component;

   iii) invalidating assumptions and properties on which lower-level components rely.

h) Executing data not intended to be executable, or making it executable includes attacks involving viruses (e.g. putting executable code or commands in a file which are automatically executed when the file is edited or accessed, thus inheriting any privileges the owner of the file has).

i) Generating unexpected input for a component can have unexpected effects which an attacker could take advantage of. For example, if the TOE is an application implementing security functions that could be bypassed if a user gains access to the underlying operating system, it may be possible to gain such access following the login sequence by exploring the effect of hitting various control or escape sequences whilst a password is being authenticated.

j) Invalidating assumptions and properties on which lower level components rely includes attacks based on breaking out of the constraints of an application to gain access to an underlying operating system in order to bypass the security functions implemented by the application. In this case the assumption being invalidated is that it is not possible for a user of the application to gain such access. A similar attack can be envisaged if security functions are implemented by an application on an underlying database management system: again the security functions could be bypassed if an attacker can break out of the constraints of the application.

k) Attacks based on reading sensitive data stored in inadequately protected areas (applicable where confidentiality is a concern) include the following issues which should be considered as possible means of gaining access to sensitive data:

i) disk scavenging;

ii) access to unprotected memory;

iii) exploiting access to shared writable files or other shared resources (e.g. swap files);

l) Activating error recovery to determine what access users can obtain. For example, after a crash an automatic file recovery system may employ a lost and found directory for headerless files, which are on disc without labels. If the TOE implements mandatory access controls, it is important to investigate at what security level this directory is kept (e.g. at system high), and who has access to this directory.

Tampering

31 Tampering includes any attack based on an attacker attempting to influence the behaviour of a security function or mechanism (i.e. corruption or de-activation), for example by:

a) accessing data on whose confidentiality or integrity the security function or mechanism relies;

b) forcing the TOE to cope with unusual or unexpected circumstances;

c) disabling or delaying security enforcement.

32 Each of the following should be considered (where relevant) in the evaluator's independent vulnerability analysis.

a) Attacks based on accessing data on whose confidentiality or integrity the security function or mechanism include:

i) reading, writing or modifying internal data directly or indirectly;

ii) using a component in an unexpected context or for an unexpected purpose;

iii) using interference between components that are not visible at a higher level of abstraction.

b) Reading, writing or modifying internal data directly or indirectly includes the following types of attack which should be considered:

i) reading 'secrets' stored internally, such as user passwords;

ii) spoofing internal data that security enforcing mechanisms rely upon;

iii) modifying environment variables (e.g. logical names), or data in configuration files or temporary files.

c) It may be possible to hoodwink a trusted process into modifying a protected file that it wouldn't normally access.

d) The evaluator should also consider the following 'dangerous features':

i) source code resident on the TOE along with a compiler (for instance, it may be possible to modify the login source code);

ii) an interactive debugger and patch facility (for instance, it may be possible to modify the executable image);

iii) the possibility of making changes at device controller level, where file protection does not exist;

iv) diagnostic code which exists in the source code and that may be optionally included;

v) developer's tools left in the TOE.

e) Using a component in an unexpected context or for an unexpected purpose includes (for example), where the TOE is an application built upon an operating system, users exploiting knowledge of a word processor package or other editor to modify their own command file (e.g. to acquire greater privileges).

f) Using interference between components which are not visible at a higher level of abstraction includes attacks exploiting shared access to resources, where modification of a resource by one component can influence the behaviour of another (trusted) component, e.g. at source code level, through the use of global data or indirect mechanisms such as shared memory or semaphores.

g) Attacks based on forcing the TOE to cope with unusual or unexpected circumstances should always be considered. Relevant items include:

i) generating unexpected input for a component;

ii) invalidating assumptions and properties on which lower-level components rely.

h) Generating unexpected input for a component includes investigating the behaviour of the TOE when:

i) command input buffers overflow (possibly 'crashing the stack' or overwriting other storage, which an attacker may be able to take advantage of, or forcing a crash dump that may contain sensitive information such as clear text passwords);

ii) invalid commands or parameters are entered (including supplying a read only parameter to an interface which expects to return data via that parameter);

iii) an end-of-file marker (e.g. CTRL/Z or CTRL/D) or null character is inserted in an audit trail.

i) Invalidating assumptions and properties on which lower-level components rely includes attacks taking advantage of errors in the source code where the code assumes (explicitly or implicitly) that security relevant data is in a particular format or has a particular range of values. In these cases the evaluator should determine whether they can invalidate such assumptions by causing the data to be in a different format or to have different values, and if so whether this could confer advantage to an attacker.

j) The correct behaviour of the security functions may be dependent on assumptions that are invalidated under extreme circumstances where resource limits are reached or parameters reach their maximum value. The evaluator should consider (where practical) the behaviour of the TOE when these limits are reached, for example:

   i) changing dates (e.g. examining how the TOE behaves when a critical date threshold is passed);

   ii) filling discs;

   iii) exceeding the maximum number of users;

   iv) filling the audit log;

   v) saturating security alarm queues at a console;

   vi) overloading various parts of a multi-user TOE which relies heavily upon communications components;

   vii) swamping a network, or individual hosts, with traffic;

   viii) filling buffers or fields.

k) Attacks based on disabling or delaying security enforcement include the following items:

   i) using interrupts or scheduling functions to disrupt sequencing;

   ii) disrupting concurrence;

   iii) using interference between components which are not visible at a higher level of abstraction.

l) Using interrupts or scheduling functions to disrupt sequencing includes investigating the behaviour of the TOE when:

i) a command is interrupted (with CTRL/C, CTRL/Y, etc.);

ii) a second interrupt is issued before the first is acknowledged.

m) The effects of terminating security critical processes (e.g. an audit daemon) should be explored. Similarly, it may be possible to delay the logging of audit records or the issuing or receipt of alarms such that it is of no use to an administrator (since the attack may already have succeeded).

n) Disrupting concurrence includes investigating the behaviour of the TOE when two or more subjects attempt simultaneous access. It may be that the TOE can cope with the interlocking required when two subjects attempt simultaneous access, but that the behaviour becomes less well defined in the presence of further subjects. For example, a critical security process could be put into a resource-wait state if two other processes are accessing a resource which it requires.

o) Using interference between components which are not visible at a higher level of abstraction may provide a means of delaying a time-critical trusted process.

Direct attacks

33 Direct attack includes the identification of any penetration tests necessary to confirm or disprove the claimed minimum strength of functions. When identifying penetration tests under this heading, the evaluator should also be aware of the possibility of vulnerabilities existing as a result of security mechanisms being susceptible to direct attack.

Misuse

34 Misuse includes the identification of any penetration tests necessary to confirm or disprove the misuse analysis. Issues to be considered include:

a) behaviour of the TOE when start-up, closedown or error recovery is activated;

b) behaviour of the TOE under extreme circumstances (sometimes termed overload or asymptotic behaviour), particularly where this could lead to the deactivation or disabling of a security enforcing function or mechanism;

c) any potential for unintentional misconfiguration or insecure use arising from attacks noted in the section on tampering above.

1.3.4 AVA_VLA.4.4E

AVA_VLA.4.4E	The evaluator shall perform independent penetration testing, based on the independent vulnerability analysis, to determine the exploitability of additional identified vulnerabilities in the intended environment.

AVA_VLA.4-12 The evaluator ***shall devise*** penetration tests, based on the independent vulnerability analysis.

35 The evaluator prepares for penetration testing based on the prioritised list of vulnerabilities hypothesised in evaluator action AVA_VLA.4.3E.

36 With an understanding of the suspected vulnerability, the evaluator determines the most feasible way to test for the TOE's susceptibility. Specifically the evaluator considers:

a) the security function interfaces that will be used to stimulate the TSF and observe responses;

b) initial conditions that will need to exist for the test (i.e. any particular objects or subjects that will need to exist and security attributes they will need to have);

c) special test equipment that will be required to either stimulate a security function or make observations of a security function.

37 The evaluator will probably find it practical to carry out penetration test using a series of test cases, where each test case will test for a specific vulnerability.

AVA_VLA.4-13 The evaluator ***shall produce*** penetration test documentation for the tests based on the independent vulnerability analysis, in sufficient detail to enable the tests to be repeatable. The test documentation shall include:

a) identification of the obvious vulnerability the TOE is being tested for;

b) instructions to connect and set-up all required test equipment as required to conduct the penetration test;

c) instructions to establish all penetration test prerequisite initial conditions;

d) instructions to stimulate the TSF;

e) instructions for observing the behaviour of the TSF;

f) descriptions of all expected results and the necessary analysis to be performed on the observed behaviour for comparison against expected results;

g) instructions to conclude the test and establish the necessary post-test state for the TOE.

38 The intent of specifying this level of detail in the test documentation is to allow another evaluator to repeat the tests and obtain an equivalent result.

AVA_VLA.4-14 The evaluator ***shall conduct*** penetration testing, based on the independent vulnerability analysis.

39 The evaluator uses the penetration test documentation resulting from work unit AVA_VLA.4-12 as a basis for executing penetration tests on the TOE, but this does not preclude the evaluator from performing additional ad hoc penetration tests. If required, the evaluator may devise new tests as a result of information learned during penetration testing that, if performed by the evaluator, are to be recorded in the penetration test documentation. Such tests may be required to follow up unexpected results or observations, or to investigate potential vulnerabilities suggested to the evaluator during the pre-planned testing.

40 Should penetration testing show that a hypothesised vulnerability does not exist, then the evaluator should determine whether or not the evaluator's own analysis was incorrect, or if evaluation deliverables are incorrect or incomplete.

AVA_VLA.4-15 The evaluator ***shall record*** the actual results of the penetration tests.

41 While some specific details of the actual test results may be different from those expected (e.g. time and date fields in an audit record) the overall result should be identical. Any differences should be justified.

AVA_VLA.4-16 The evaluator ***shall report*** in the ETR the evaluator penetration testing effort, outlining the testing approach, configuration, depth and results.

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

43 Information that would typically be found in the ETR section regarding evaluator penetration testing efforts is:

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

b) security functions penetration tested. A brief listing of the security functions that were the focus of the penetration testing;

c) verdict for the sub-activity. The overall judgement on the results of penetration testing.

44 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 penetration testing the evaluator performed during the evaluation.

1.3.5 AVA_VLA.4.5E

AVA_VLA.4.5E	The evaluator shall determine that the TOE is resistant to penetration attacks performed by an attacker possessing a **high** attack potential.

AVA_VLA.4-17 The evaluator ***shall examine*** the results of all penetration testing and the conclusions of all vulnerability analysis to determine that the TOE, in its intended environment, is resistant to an attacker possessing a high attack potential.

45 If the results reveal that the TOE, in its intended environment, has vulnerabilities exploitable by an attacker possessing a high attack potential or less, then this evaluator action fails.

AVA_VLA.4-18 The evaluator shall report in the ETR all exploitable vulnerabilities and residual vulnerabilities, detailing for each:

a) its source (e.g. CEM activity being undertaken when it was conceived, known to the evaluator, read in a publication);

b) the implicated security function(s), objective(s) not met, organisational security policy(ies) contravened and threat(s) realised;

c) a description;

d) whether it is exploitable in its intended environment or not (i.e. exploitable or residual);

e) identification of evaluation party (e.g. developer, evaluator) who identified it.