Australian Government
Department of Defence

# ASD-Approved Supporting Document

## Supporting Document
## Mandatory Technical Document
## Evaluation Activities for Network Device cPP

| | |
|---|---|
| Version: | **1.0** |
| Technology type: | **Network Devices** |
| Authored by: | **Common Criteria** |
| Publication date: | **February 2015** |
| ASD approval date: | **30 May 2016** |

*The following document is a Supporting Document (SD) authored by Common Criteria. This Supporting Document has been approved for use by the Australian Signals Directorate.*

*This Mandatory Technical Document supports the collaborative Protection Profile for Network Devices (ND cPP).*

*This SD has been developed by Network Devices iTC and is designed to be used to support the evaluations of products.*

*For information relating to the application of Protection Profiles, please refer to the Australian Government Information Security Manual (ISM) or www.asd.gov.au Controls in the ISM take precedence over any requirements contained in this SP where there is a conflict.*


# Supporting Document
# Mandatory Technical Document

---

Evaluation Activities for Network Device
cPP

February-2015

Version 1.0

CCDB-2015-01-001

# Foreword

This is a supporting document, intended to complement the Common Criteria version 3 and the associated Common Evaluation Methodology for Information Technology Security Evaluation.

Supporting documents may be "Guidance Documents", that highlight specific approaches and application of the standard to areas where no mutual recognition of its application is required, and as such, are not of normative nature, or "Mandatory Technical Documents", whose application is mandatory for evaluations whose scope is covered by that of the supporting document. The usage of the latter class is not only mandatory, but certificates issued as a result of their application are recognized under the CCRA.

This supporting document has been developed by the Network International Technical Community (NDFW-iTC) and is designed to be used to support the evaluations of products against the cPPs identified in section 1.1.

**Technical Editor:** Network International Technical Community (NDFW-iTC)

**Document history:**

V1.0, 27 February 2015 (published version)

V0.4, 26 January 2015 (incorporates changes due to comments received from CCDB review)

V0.3, 17 October 2014 (released version following public review, submitted for CCDB review)

V0.2, 13 October 2014 (internal draft in response to public review comments, for iTC review)

V0.1, 5 September 2014 (Initial release for public review)

**General Purpose:** See section 1.1.

**Field of special use:** This Supporting Document applies to the evaluation of TOEs claiming conformance with the collaborative Protection Profile for Network Devices [NDcPP] and collaborative Protection Profile for Stateful Traffic Filter Firewalls [FWcPP].

**Acknowledgements:**

This Supporting Document was developed by the Network international Technical Community with representatives from industry, Government agencies, Common Criteria Test Laboratories, and members of academia.

# Table of Contents

## Table of contents

# List of tables

# 1 Introduction

## 1.1 Technology Area and Scope of Supporting Document

1 This Supporting Document defines the Evaluation Activities associated with the collaborative Protection Profile for Network Devices [NDcPP].

2 The Network Device technical area has a number of specialised aspects, such as those relating to the secure implementation and use of protocols, and to the particular ways in which remote management facilities need to be assessed across a range of different physical and logical interfaces for different types of infrastructure devices. This degree of specialisation, and the associations between individual SFRs in the cPP, make it important for both efficiency and effectiveness that evaluation activities are given more specific interpretations than those found in the generic CEM activities.

3 This Supporting Document is mandatory for evaluations of products that claim conformance to any of the following cPP(s):

a) collaborative Protection Profile for Network Devices [NDcPP]

b) collaborative Protection Profile for Stateful Traffic Filter Firewalls [FWcPP].

4 Although Evaluation Activities are defined mainly for the evaluators to follow, the definitions in this Supporting Document aim to provide a common understanding for developers, evaluators and users as to what aspects of the TOE are tested in an evaluation against the associated cPPs, and to what depth the testing is carried out. This common understanding in turn contributes to the goal of ensuring that evaluations against the cPP achieve comparable, transparent and repeatable results. In general the definition of Evaluation Activities will also help Developers to prepare for evaluation by identifying specific requirements for their TOE. The specific requirements in Evaluation Activities may in some cases clarify the meaning of SFRs, and may identify particular requirements for the content of Security Targets (especially the TOE Summary Specification), user guidance documentation, and possibly supplementary information (e.g. for entropy analysis or cryptographic key management architecture – see section 4).

## 1.2 Structure of the Document

5 Evaluation Activities can be defined for both Security Functional Requirements and Security Assurance Requirements. These are defined in separate sections of this Supporting Document.

6 If any Evaluation Activity cannot be successfully completed in an evaluation then the overall verdict for the evaluation is a ‘fail’. In rare cases there may be acceptable reasons why an Evaluation Activity may be modified or deemed not applicable for a particular TOE, but this must be agreed with the Certification Body for the evaluation.

7 In general, if all Evaluation Activities (for both SFRs and SARs) are successfully completed in an evaluation then it would be expected that the overall verdict for the evaluation is a 'pass'. To reach a 'fail' verdict when the Evaluation Activities have been successfully completed would require a specific justification from the evaluator as to why the Evaluation Activities were not sufficient for that TOE.

8 Similarly, at the more granular level of Assurance Components, if the Evaluation Activities for an Assurance Component and all of its related SFR Evaluation Activities are successfully completed in an evaluation then it would be expected that the verdict for the Assurance Component is a 'pass'. To reach a 'fail' verdict for the Assurance Component when these Evaluation Activities have been successfully completed would require a specific justification from the evaluator as to why the Evaluation Activities were not sufficient for that TOE.

## 1.3 Glossary

9 For definitions of standard CC terminology see [CC] part 1.

10 **cPP** – collaborative Protection Profile

11 **CVE** – Common Vulnerabilities and Exposures (database)

12 **iTC** – International Technical Community

13 **SD** – Supporting Document

14 **Supplementary information –** information that is not necessarily included in the Security Target or guidance documentation, and that may not necessarily be public. Examples of such information could be entropy analysis, or description of a cryptographic key management architecture used in (or in support of) the TOE. The requirement for any such supplementary information will be identified in the relevant cPP (see description in section 4).

# 2 Evaluation Activities for SFRs

## 2.1 Security Audit (FAU)

### 2.1.1 FAU_GEN.1 Audit data generation

#### 2.1.1.1 Guidance Documentation

15 The evaluator shall check the guidance documentation and ensure that it lists all of the auditable events and provides a format for audit records. Each audit record format type must be covered, along with a brief description of each field. The evaluator shall check to make sure that every audit event type mandated by the cPP is described and that the description of the fields contains the information required in FAU_GEN1.2, and the additional information specified in the table of audit events.

16 The evaluator shall also make a determination of the administrative actions that are relevant in the context of the cPP. The evaluator shall examine the guidance documentation and make a determination of which administrative commands, including subcommands, scripts, and configuration files, are related to the configuration (including enabling or disabling) of the mechanisms implemented in the TOE that are necessary to enforce the requirements specified in the cPP. The evaluator shall document the methodology or approach taken while determining which actions in the administrative guide are security relevant with respect to the cPP. The evaluator may perform this activity as part of the activities associated with ensuring that the corresponding guidance documentation satisfies the requirements related to it.

#### 2.1.1.2 Tests

17 The evaluator shall test the TOE's ability to correctly generate audit records by having the TOE generate audit records for the events listed in the table of audit events and administrative actions listed above. This should include all instances of an event: for instance, if there are several different I&A mechanisms for a system, the FIA_UIA_EXT.1 events must be generated for each mechanism. The evaluator shall test that audit records are generated for the establishment and termination of a channel for each of the cryptographic protocols contained in the ST. If HTTPS is implemented, the test demonstrating the establishment and termination of a TLS session can be combined with the test for an HTTPS session. Logging of all activities related to trusted update should be tested in detail and with utmost diligence. When verifying the test results, the evaluator shall ensure the audit records generated during testing match the format specified in the guidance documentation, and that the fields in each audit record have the proper entries.

18 Note that the testing here can be accomplished in conjunction with the testing of the security mechanisms directly.

### 2.1.2 FAU_GEN.2 User identity association

19 This activity should be accomplished in conjunction with the testing of FAU_GEN.1.1.

### 2.1.3 FAU_STG.1 Protected audit trail storage

#### 2.1.3.1 TSS

20 The evaluator shall examine the TSS to ensure it describes the amount of audit data that are stored locally and how these records are protected against unauthorized modification or deletion. The evaluator shall ensure that the TSS describes the conditions that must be met for authorized deletion of audit records.

#### 2.1.3.2 Guidance Documentation

21 The evaluator shall examine the guidance documentation to determine that it describes any configuration required for protection of the locally stored audit data against unauthorized modification or deletion.

#### 2.1.3.3 Tests

22 The evaluator shall perform the following tests:

a) Test 1: The evaluator shall access the audit trail as an *unauthorized* administrator and attempt to modify and delete the audit records. The evaluator shall verify that these attempts fail.

b) Test 2: The evaluator shall access the audit trail as an authorized administrator and attempt to delete the audit records. The evaluator shall verify that these attempts succeed. The evaluator shall verify that only the records authorized for deletion are deleted.

### 2.1.4 FAU_STG_EXT.1 Protected audit event storage

#### 2.1.4.1 TSS

23 The evaluator shall examine the TSS to ensure it describes the means by which the audit data are transferred to the external audit server, and how the trusted channel is provided.

24 The evaluator shall examine the TSS to ensure it describes the amount of audit data that are stored locally; what happens when the local audit data store is full; and how these records are protected against unauthorized access.

25 If the TOE complies with FAU_STG_EXT.2 the evaluator shall verify that the numbers provided by the TOE according to the selection for FAU_STG_EXT.2 are correct when performing the tests for FAU_STG_EXT.1.3.

26 The evaluator shall examine the TSS to ensure that it details the behaviour of the TOE when the storage space for audit data is full. When the option 'overwrite previous audit record' is selected this description should include an outline of the rule for overwriting audit data. If 'other actions' are chosen such as sending the new audit data to an external IT entity, then the related behaviour of the TOE shall also be detailed in the TSS.

#### 2.1.4.2 Guidance Documentation

27 The evaluator shall also examine the guidance documentation to ensure it describes how to establish the trusted channel to the audit server, as well as describe any requirements on the audit server (particular audit server protocol, version of the protocol required, etc.), as well as configuration of the TOE needed to communicate with the audit server.

28 The evaluator shall also examine the guidance documentation to determine that it describes the relationship between the local audit data and the audit data that are sent to the audit log server. For example, when an audit event is generated, is it simultaneously sent to the external server and the local store, or is the local store used as a buffer and "cleared" periodically by sending the data to the audit server.

29 The evaluator shall also ensure that the guidance documentation describes all possible configuration options for FAU_STG_EXT.1.3 and the resulting behaviour of the TOE for each possible configuration. The description of possible configuration options and resulting behaviour shall correspond to those described in the TSS.

#### 2.1.4.3 Tests

30 Testing of the trusted channel mechanism for audit will be performed as specified in the associated assurance activities for the particular trusted channel mechanism. The evaluator shall perform the following additional test for this requirement:

a) Test 1: The evaluator shall establish a session between the TOE and the audit server according to the configuration guidance provided. The evaluator shall then examine the traffic that passes between the audit server and the TOE during several activities of the evaluator's choice designed to generate audit data to be transferred to the audit server. The evaluator shall observe that these data are not able to be viewed in the clear during this transfer, and that they are successfully received by the audit server. The evaluator shall record the particular software (name, version) used on the audit server during testing.

31 The evaluator shall perform operations that generate audit data and verify that this data is stored locally. The evaluator shall perform operations that generate audit data until the local storage space is exceeded and verifies that the TOE complies with the behaviour defined in FAU_STG_EXT.1.3. Depending on the configuration this means that the evaluator has to check

the content of the audit data when the audit data is just filled to the maximum and then verifies that

a) The audit data remains unchanged with every new auditable event that should be tracked but that the audit data is recorded again after the local storage for audit data is cleared (for the option ‘drop new audit data’ in FAU_STG_EXT.1.3).

b) The existing audit data is overwritten with every new auditable event that should be tracked according to the specified rule (for the option ‘overwrite previous audit records’ in FAU_STG_EXT.1.3)

c) The TOE behaves as specified (for the option ‘other action’ in FAU_STG_EXT.1.3).

### 2.1.5 FAU_ STG_EXT.2 Counting lost audit data

32 This activity should be accomplished in conjunction with the testing of FAU_STG_EXT.1.2 and FAU_STG_EXT.1.3.

#### 2.1.5.1 TSS

33 The evaluator shall examine the TSS to ensure that it details the possible options the TOE supports for information about the number of audit records that have been dropped, overwritten, etc. if the local storage for audit data is full.

#### 2.1.5.2 Guidance Documentation

34 The evaluator shall also ensure that the guidance documentation describes all possible configuration options and the meaning of the result returned by the TOE for each possible configuration. The description of possible configuration options and explanation of the result shall correspond to those described in the TSS.

35 The evaluator shall verify that the guidance documentation contains a warning for the administrator about the loss of audit data when he clears the local storage for audit records.

#### 2.1.5.3 Tests

36 The evaluator shall verify that the numbers provided by the TOE according to the selection for FAU_STG_EXT.2 are correct when performing the tests for FAU_STG_EXT.1.3.

### 2.1.6 FAU_ STG_EXT.3 Display warning for local storage space

37 This activity should be accomplished in conjunction with the testing of FAU_STG_EXT.1.2 and FAU_STG_EXT.1.3.

#### 2.1.6.1 TSS

38 The evaluator shall examine the TSS to ensure that it details how the user is warned before the local storage for audit data is full.

#### 2.1.6.2 Guidance Documentation

39 The evaluator shall also ensure that the guidance documentation describes how the user is warned before the local storage for audit data is full and how this warning is displayed or stored (since there is no guarantee that an administrator session is running at the time the warning is issued, it is probably stored in the log files). The description in the guidance documentation shall correspond to the description in the TSS.

#### 2.1.6.3 Tests

40 The evaluator shall verify that a warning is issued by the TOE before the local storage space for audit data is full.

## 2.2 Cryptographic Support (FCS)

### 2.2.1 FCS_CKM.1 Cryptographic Key Generation

#### 2.2.1.1 TSS

41 The evaluator shall ensure that the TSS identifies the key sizes supported by the TOE. If the ST specifies more than one scheme, the evaluator shall examine the TSS to verify that it identifies the usage for each scheme.

#### 2.2.1.2 Guidance Documentation

42 The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected key generation scheme(s) and key size(s) for all uses defined in this PP.

#### 2.2.1.3 Tests

43 Note: The following tests require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on factory products.

**Key Generation for FIPS PUB 186-4 RSA Schemes**

44 The evaluator shall verify the implementation of RSA Key Generation by the TOE using the Key Generation test. This test verifies the ability of the TSF to correctly produce values for the key components including the public verification exponent *e*, the private prime factors *p* and *q*, the public modulus *n* and the calculation of the private signature exponent *d*.

45 Key Pair generation specifies 5 ways (or methods) to generate the primes *p* and *q*. These include:

a) Random Primes:

- Provable primes
- Probable primes

b) Primes with Conditions:

- Primes p1, p2, q1,q2, p and q shall all be provable primes
- Primes p1, p2, q1, and q2 shall be provable primes and p and q shall be probable primes
- Primes p1, p2, q1,q2, p and q shall all be probable primes

46 To test the key generation method for the Random Provable primes method and for all the Primes with Conditions methods, the evaluator must seed the TSF key generation routine with sufficient data to deterministically generate the RSA key pair. This includes the random seed(s), the public exponent of the RSA key, and the desired key length. For each key length supported, the evaluator shall have the TSF generate 25 key pairs. The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated from a known good implementation.

***Key Generation for Elliptic Curve Cryptography (ECC)***

*FIPS 186-4 ECC Key Generation Test*

47 For each supported NIST curve, i.e., P-256, P-384 and P-521, the evaluator shall require the implementation under test (IUT) to generate 10 private/public key pairs. The private key shall be generated using an approved random bit generator (RBG). To determine correctness, the evaluator shall submit the generated key pairs to the public key verification (PKV) function of a known good implementation.

*FIPS 186-4 Public Key Verification (PKV) Test*

48 For each supported NIST curve, i.e., P-256, P-384 and P-521, the evaluator shall generate 10 private/public key pairs using the key generation function of a known good implementation and modify five of the public key values so that they are incorrect, leaving five values unchanged (i.e., correct). The evaluator shall obtain in response a set of 10 PASS/FAIL values.

***Key Generation for Finite-Field Cryptography (FFC)***

49 The evaluator shall verify the implementation of the Parameters Generation and the Key Generation for FFC by the TOE using the Parameter Generation and Key Generation test. This test verifies the ability of the TSF to correctly produce values for the field prime p, the cryptographic prime q (dividing p-1), the cryptographic group generator g, and the calculation of the private key x and public key y.

50 The Parameter generation specifies 2 ways (or methods) to generate the cryptographic prime q and the field prime p:

- Primes q and p shall both be provable primes
- Primes q and field prime p shall both be probable primes

51 and two ways to generate the cryptographic group generator g:

- Generator g constructed through a verifiable process
- Generator g constructed through an unverifiable process.

52 The Key generation specifies 2 ways to generate the private key x:

- len(q) bit output of RBG where 1 <=x <= q-1
- len(q) + 64 bit output of RBG, followed by a mod q-1 operation where 1<= x<=q-1.

53 The security strength of the RBG must be at least that of the security offered by the FFC parameter set.

54 To test the cryptographic and field prime generation method for the provable primes method and/or the group generator g for a verifiable process, the evaluator must seed the TSF parameter generation routine with sufficient data to deterministically generate the parameter set.

55 For each key length supported, the evaluator shall have the TSF generate 25 parameter sets and key pairs. The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated from a known good implementation. Verification must also confirm

- g != 0,1
- q divides p-1
- g^q mod p = 1
- g^x mod p = y

56 for each FFC parameter set and key pair.

### 2.2.2 FCS_CKM.2 Cryptographic Key Establishment

#### 2.2.2.1 TSS

57 The evaluator shall ensure that the supported key establishment schemes correspond to the key generation schemes identified in FCS_CKM.1.1. If the ST specifies more than one scheme, the evaluator shall examine the TSS to verify that it identifies the usage for each scheme.

#### 2.2.2.2 Guidance Documentation

58 The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected key establishment scheme(s).

#### 2.2.2.3 Tests

**Key Establishment Schemes**

59 The evaluator shall verify the implementation of the key establishment schemes of the supported by the TOE using the applicable tests below.

***SP800-56A Key Establishment Schemes***

60 The evaluator shall verify a TOE's implementation of SP800-56A key agreement schemes using the following Function and Validity tests. These validation tests for each key agreement scheme verify that a TOE has implemented the components of the key agreement scheme according to the specifications in the Recommendation. These components include the calculation of the DLC primitives (the shared secret value Z) and the calculation of the derived keying material (DKM) via the Key Derivation Function (KDF). If key confirmation is supported, the evaluator shall also verify that the components of key confirmation have been implemented correctly, using the test procedures described below. This includes the parsing of the DKM, the generation of MACdata and the calculation of MACtag.

*Function Test*

61 The Function test verifies the ability of the TOE to implement the key agreement schemes correctly. To conduct this test the evaluator shall generate or obtain test vectors from a known good implementation of the TOE supported schemes. For each supported key agreement scheme-key agreement role combination, KDF type, and, if supported, key confirmation role- key confirmation type combination, the tester shall generate 10 sets of test vectors. The data set consists of one set of domain parameter values (FFC) or the NIST approved curve (ECC) per 10 sets of public keys. These keys are static, ephemeral or both depending on the scheme being tested.

62 The evaluator shall obtain the DKM, the corresponding TOE's public keys (static and/or ephemeral), the MAC tag(s), and any inputs used in the KDF, such as the Other Information field OI and TOE id fields.

63 If the TOE does not use a KDF defined in SP 800-56A, the evaluator shall obtain only the public keys and the hashed value of the shared secret.

64 The evaluator shall verify the correctness of the TSF's implementation of a given scheme by using a known good implementation to calculate the shared secret value, derive the keying material DKM, and compare hashes or MAC tags generated from these values.

65 If key confirmation is supported, the TSF shall perform the above for each implemented approved MAC algorithm.

*Validity Test*

66 The Validity test verifies the ability of the TOE to recognize another party's valid and invalid key agreement results with or without key confirmation. To conduct this test, the evaluator shall obtain a list of the supporting cryptographic functions included in the SP800-56A key agreement implementation to determine which errors the TOE should be able to recognize. The evaluator generates a set of 24 (FFC) or 30 (ECC) test vectors consisting of data sets including domain parameter values or NIST approved curves, the evaluator's public keys, the TOE's public/private key pairs, MACTag, and any inputs used in the KDF, such as the other info and TOE id fields.

67 The evaluator shall inject an error in some of the test vectors to test that the TOE recognizes invalid key agreement results caused by the following fields being incorrect: the shared secret value Z, the DKM, the other information field OI, the data to be MACed, or the generated MACTag. If the TOE contains the full or partial (only ECC) public key validation, the evaluator will also individually inject errors in both parties' static public keys, both parties' ephemeral public keys and the TOE's static private key to assure the TOE detects errors in the public key validation function and/or the partial key validation function (in ECC only). At least two of the test vectors shall remain unmodified and therefore should result in valid key agreement results (they should pass).

68 The TOE shall use these modified test vectors to emulate the key agreement scheme using the corresponding parameters. The evaluator shall compare the TOE's results with the results using a known good implementation verifying that the TOE detects these errors.

***SP800-56B Key Establishment Schemes***

69 The evaluator shall verify that the TSS describes whether the TOE acts as a sender, a recipient, or both for RSA-based key establishment schemes.

70 If the TOE acts as a sender, the following assurance activity shall be performed to ensure the proper operation of every TOE supported combination of RSA-based key establishment scheme:

a) To conduct this test the evaluator shall generate or obtain test vectors from a known good implementation of the TOE supported schemes. For each combination of supported key establishment scheme and its options (with or without key confirmation if supported, for each supported key confirmation MAC function if key confirmation is supported, and for each supported mask generation function if KTS-OAEP is supported), the tester shall generate 10 sets of test vectors. Each test vector shall include the RSA public key, the plaintext keying material, any additional input parameters if applicable, the MacKey and MacTag if key confirmation is incorporated, and the outputted ciphertext. For each test vector, the evaluator shall perform a key establishment encryption operation on the TOE with the same inputs (in cases where key confirmation is incorporated, the test shall

use the MacKey from the test vector instead of the randomly generated MacKey used in normal operation) and ensure that the outputted ciphertext is equivalent to the ciphertext in the test vector.

71 If the TOE acts as a receiver, the following assurance activities shall be performed to ensure the proper operation of every TOE supported combination of RSA-based key establishment scheme:

a) To conduct this test the evaluator shall generate or obtain test vectors from a known good implementation of the TOE supported schemes. For each combination of supported key establishment scheme and its options (with our without key confirmation if supported, for each supported key confirmation MAC function if key confirmation is supported, and for each supported mask generation function if KTS-OAEP is supported), the tester shall generate 10 sets of test vectors. Each test vector shall include the RSA private key, the plaintext keying material (KeyData), any additional input parameters if applicable, the MacTag in cases where key confirmation is incorporated, and the outputted ciphertext. For each test vector, the evaluator shall perform the key establishment decryption operation on the TOE and ensure that the outputted plaintext keying material (KeyData) is equivalent to the plaintext keying material in the test vector. In cases where key confirmation is incorporated, the evaluator shall perform the key confirmation steps and ensure that the outputted MacTag is equivalent to the MacTag in the test vector.

b) The evaluator shall ensure that the TSS describes how the TOE handles decryption errors. In accordance with NIST Special Publication 800-56B, the TOE must not reveal the particular error that occurred, either through the contents of any outputted or logged error message or through timing variations. If KTS-OAEP is supported, the evaluator shall create separate contrived ciphertext values that trigger each of the three decryption error checks described in NIST Special Publication 800-56B section 7.2.2.3, ensure that each decryption attempt results in an error, and ensure that any outputted or logged error message is identical for each. If KTS-KEM-KWS is supported, the evaluator shall create separate contrived ciphertext values that trigger each of the three decryption error checks described in NIST Special Publication 800-56B section 7.2.3.3, ensure that each decryption attempt results in an error, and ensure that any outputted or logged error message is identical for each.

### 2.2.3 FCS_CKM.4 Cryptographic Key Destruction

#### 2.2.3.1 TSS

72 The evaluator shall check to ensure the TSS lists each type of plaintext key material and its origin and storage location.

73 The evaluator shall verify that the TSS describes when each type of key material is cleared (for example, on system power off, on wipe function, on disconnection of trusted channels, when no longer needed by the trusted channel per the protocol, etc.).

74 The evaluator shall also verify that, for each type of key, the type of clearing procedure that is performed (cryptographic erase, overwrite with zeros, overwrite with random pattern, or block erase) is listed. If different types of memory are used to store the materials to be protected, the evaluator shall check to ensure that the TSS describes the clearing procedure in terms of the memory in which the data are stored (for example, "secret keys stored on flash are cleared by overwriting once with zeros, while secret keys stored on the internal persistent storage device are cleared by overwriting three times with a random pattern that is changed before each write").

### 2.2.4 FCS_COP.1(1) Cryptographic Operation (AES Data Encyption/ Decryption)

#### 2.2.4.1 Tests

**AES-CBC Known Answer Tests**

75 There are four Known Answer Tests (KATs), described below. In all KATs, the plaintext, ciphertext, and IV values shall be 128-bit blocks. The results from each test may either be obtained by the evaluator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

76 **KAT-1.** To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of 10 plaintext values and obtain the ciphertext value that results from AES-CBC encryption of the given plaintext using a key value of all zeros and an IV of all zeros. Five plaintext values shall be encrypted with a 128-bit all-zeros key, and the other five shall be encrypted with a 256-bit all-zeros key.

77 To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using 10 ciphertext values as input and AES-CBC decryption.

78 **KAT-2.** To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of 10 key values and obtain the ciphertext value that results from AES-CBC encryption of an all-zeros plaintext using the given key value and an IV of all zeros. Five of the keys shall be 128-bit keys, and the other five shall be 256-bit keys.

79 To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using an all-zero ciphertext value as input and AES-CBC decryption.

80 **KAT-3.** To test the encrypt functionality of AES-CBC, the evaluator shall supply the two sets of key values described below and obtain the ciphertext value that results from AES encryption of an all-zeros plaintext using the given key value and an IV of all zeros. The first set of keys shall have 128 128-bit keys, and the second set shall have 256 256-bit keys. Key *i* in each set shall have the leftmost *i* bits be ones and the rightmost *N-i* bits be zeros, for *i* in [1,N].

81 To test the decrypt functionality of AES-CBC, the evaluator shall supply the two sets of key and ciphertext value pairs described below and obtain the plaintext value that results from AES-CBC decryption of the given ciphertext using the given key and an IV of all zeros. The first set of key/ciphertext pairs shall have 128 128-bit key/ciphertext pairs, and the second set of key/ciphertext pairs shall have 256 256-bit key/ciphertext pairs. Key i in each set shall have the leftmost i bits be ones and the rightmost N-i bits be zeros, for i in [1,N]. The ciphertext value in each pair shall be the value that results in an all-zeros plaintext when decrypted with its corresponding key.

82 **KAT-4.** To test the encrypt functionality of AES-CBC, the evaluator shall supply the set of 128 plaintext values described below and obtain the two ciphertext values that result from AES-CBC encryption of the given plaintext using a 128-bit key value of all zeros with an IV of all zeros and using a 256-bit key value of all zeros with an IV of all zeros, respectively. Plaintext value i in each set shall have the leftmost i bits be ones and the rightmost 128-i bits be zeros, for i in [1,128].

83 To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using ciphertext values of the same form as the plaintext in the encrypt test as input and AES-CBC decryption.

**AES-CBC Multi-Block Message Test**

84 The evaluator shall test the encrypt functionality by encrypting an *i*-block message where $1 < i <= 10$. The evaluator shall choose a key, an IV and plaintext message of length *i* blocks and encrypt the message, using the mode to be tested, with the chosen key and IV. The ciphertext shall be compared to the result of encrypting the same plaintext message with the same key and IV using a known good implementation.

85 The evaluator shall also test the decrypt functionality for each mode by decrypting an *i*-block message where $1 < i <= 10$. The evaluator shall choose

a key, an IV and a ciphertext message of length *i* blocks and decrypt the message, using the mode to be tested, with the chosen key and IV. The plaintext shall be compared to the result of decrypting the same ciphertext message with the same key and IV using a known good implementation.

**AES-CBC Monte Carlo Tests**

86 The evaluator shall test the encrypt functionality using a set of 200 plaintext, IV, and key 3-tuples. 100 of these shall use 128 bit keys, and 100 shall use 256 bit keys. The plaintext and IV values shall be 128-bit blocks. For each 3-tuple, 1000 iterations shall be run as follows:

```
# Input: PT, IV, Key
for i = 1 to 1000:
        if i == 1:
                CT[1] = AES-CBC-Encrypt(Key, IV, PT)
                PT = IV
        else:
                CT[i] = AES-CBC-Encrypt(Key, PT)
                PT = CT[i-1]
```

87 The ciphertext computed in the 1000th iteration (i.e., CT[1000]) is the result for that trial. This result shall be compared to the result of running 1000 iterations with the same values using a known good implementation.

88 The evaluator shall test the decrypt functionality using the same test as for encrypt, exchanging CT and PT and replacing AES-CBC-Encrypt with AES-CBC-Decrypt.

**AES-GCM Test**

89 The evaluator shall test the authenticated encrypt functionality of AES-GCM for each combination of the following input parameter lengths:

***128 bit and 256 bit keys***

a) **Two plaintext lengths**. One of the plaintext lengths shall be a non-zero integer multiple of 128 bits, if supported. The other plaintext length shall not be an integer multiple of 128 bits, if supported.

b) **Three AAD lengths**. One AAD length shall be 0, if supported. One AAD length shall be a non-zero integer multiple of 128 bits, if supported. One AAD length shall not be an integer multiple of 128 bits, if supported.

c) **Two IV lengths**. If 96 bit IV is supported, 96 bits shall be one of the two IV lengths tested.

90 The evaluator shall test the encrypt functionality using a set of 10 key, plaintext, AAD, and IV tuples for each combination of parameter lengths

above and obtain the ciphertext value and tag that results from AES-GCM authenticated encrypt. Each supported tag length shall be tested at least once per set of 10. The IV value may be supplied by the evaluator or the implementation being tested, as long as it is known.

91 The evaluator shall test the decrypt functionality using a set of 10 key, ciphertext, tag, AAD, and IV 5-tuples for each combination of parameter lengths above and obtain a Pass/Fail result on authentication and the decrypted plaintext if Pass. The set shall include five tuples that Pass and five that Fail.

92 The results from each test may either be obtained by the evaluator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

### 2.2.5 FCS_COP.1(2) Cryptographic Operation (Signature Generation and Verification

#### 2.2.5.1 Tests

**ECDSA Algorithm Tests**

***ECDSA FIPS 186-4 Signature Generation Test***

93 For each supported NIST curve (i.e., P-256, P-384 and P-521) and SHA function pair, the evaluator shall generate 10 1024-bit long messages and obtain for each message a public key and the resulting signature values R and S. To determine correctness, the evaluator shall use the signature verification function of a known good implementation.

***ECDSA FIPS 186-4 Signature Verification Test***

94 For each supported NIST curve (i.e., P-256, P-384 and P-521) and SHA function pair, the evaluator shall generate a set of 10 1024-bit message, public key and signature tuples and modify one of the values (message, public key or signature) in five of the 10 tuples. The evaluator shall obtain in response a set of 10 PASS/FAIL values.

**RSA Signature Algorithm Tests**

***Signature Generation Test***

95 The evaluator shall verify the implementation of RSA Signature Generation by the TOE using the Signature Generation Test. To conduct this test the evaluator must generate or obtain 10 messages from a trusted reference implementation for each modulus size/SHA combination supported by the TSF. The evaluator shall have the TOE use their private key and modulus value to sign these messages.

96 The evaluator shall verify the correctness of the TSF's signature using a known good implementation and the associated public keys to verify the signatures.

***Signature Verification Test***

97 The evaluator shall perform the Signature Verification test to verify the ability of the TOE to recognize another party's valid and invalid signatures. The evaluator shall inject errors into the test vectors produced during the Signature Verification Test by introducing errors in some of the public keys e, messages, IR format, and/or signatures. The TOE attempts to verify the signatures and returns success or failure.

98 The evaluator shall use these test vectors to emulate the signature verification test using the corresponding parameters and verify that the TOE detects these errors.

## 2.2.6 FCS_COP.1(3) Cryptographic Operation (Hash Algorithm)

### 2.2.6.1 TSS

99 The evaluator shall check that the association of the hash function with other TSF cryptographic functions (for example, the digital signature verification function) is documented in the TSS.

### 2.2.6.2 Guidance Documentation

100 The evaluator checks the AGD documents to determine that any configuration that is required to configure the required hash sizes is present.

### 2.2.6.3 Tests

101 The TSF hashing functions can be implemented in one of two modes. The first mode is the byte-oriented mode. In this mode the TSF only hashes messages that are an integral number of bytes in length; i.e., the length (in bits) of the message to be hashed is divisible by 8. The second mode is the bit-oriented mode. In this mode the TSF hashes messages of arbitrary length. As there are different tests for each mode, an indication is given in the following sections for the bit-oriented vs. the byte-oriented testmacs.

102 The evaluator shall perform all of the following tests for each hash algorithm implemented by the TSF and used to satisfy the requirements of this PP.

**Short Messages Test - Bit-oriented Mode**

103 The evaluators devise an input set consisting of m+1 messages, where m is the block length of the hash algorithm. The length of the messages range

sequentially from 0 to m bits. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

**Short Messages Test - Byte-oriented Mode**

104 The evaluators devise an input set consisting of m/8+1 messages, where m is the block length of the hash algorithm. The length of the messages range sequentially from 0 to m/8 bytes, with each message being an integral number of bytes. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

**Selected Long Messages Test - Bit-oriented Mode**

105 The evaluators devise an input set consisting of m messages, where m is the block length of the hash algorithm (e.g. 512 bits for SHA-256). The length of the ith message is $m + 99*i$, where $1 \leq i \leq m$. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

**Selected Long Messages Test - Byte-oriented Mode**

106 The evaluators devise an input set consisting of m/8 messages, where m is the block length of the hash algorithm (e.g. 512 bits for SHA-256). The length of the ith message is $m + 8*99*i$, where $1 \leq i \leq m/8$. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

**Pseudorandomly Generated Messages Test**

107 This test is for byte-oriented implementations only. The evaluators randomly generate a seed that is n bits long, where n is the length of the message digest produced by the hash function to be tested. The evaluators then formulate a set of 100 messages and associated digests by following the algorithm provided in Figure 1 of [SHAVS]. The evaluators then ensure that the correct result is produced when the messages are provided to the TSF.

### 2.2.7 FCS_COP.1(4) Cryptographic Operation (Keyed Hash Algorithm)

#### 2.2.7.1 TSS

108 The evaluator shall examine the TSS to ensure that it specifies the following values used by the HMAC function: key length, hash function used, block size, and output MAC length used.

#### 2.2.7.2 Tests

109 For each of the supported parameter sets, the evaluator shall compose 15 sets of test data. Each set shall consist of a key and message data. The evaluator shall have the TSF generate HMAC tags for these sets of test data. The resulting MAC tags shall be compared to the result of generating HMAC tags with the same key and IV using a known good implementation.

### 2.2.8 FCS_RBG_EXT.1 Extended: Cryptographic Operation (Random Bit Generation)

110 Documentation shall be produced—and the evaluator shall perform the activities—in accordance with Appendix D of [NDcPP].

#### 2.2.8.1 Tests

111 The evaluator shall perform 15 trials for the RNG implementation. If the RNG is configurable, the evaluator shall perform 15 trials for each configuration. The evaluator shall also confirm that the guidance documentation contains appropriate instructions for configuring the RNG functionality.

112 If the RNG has prediction resistance enabled, each trial consists of (1) instantiate DRBG, (2) generate the first block of random bits (3) generate a second block of random bits (4) uninstantiate. The evaluator verifies that the second block of random bits is the expected value. The evaluator shall generate eight input values for each trial. The first is a count (0 – 14). The next three are entropy input, nonce, and personalization string for the instantiate operation. The next two are additional input and entropy input for the first call to generate. The final two are additional input and entropy input for the second call to generate. These values are randomly generated. "generate one block of random bits" means to generate random bits with number of returned bits equal to the Output Block Length (as defined in NIST SP800-90A).

113 If the RNG does not have prediction resistance, each trial consists of (1) instantiate DRBG, (2) generate the first block of random bits (3) reseed, (4) generate a second block of random bits (5) uninstantiate. The evaluator verifies that the second block of random bits is the expected value. The evaluator shall generate eight input values for each trial. The first is a count (0 – 14). The next three are entropy input, nonce, and personalization string for the instantiate operation. The fifth value is additional input to the first call to generate. The sixth and seventh are additional input and entropy input to the call to reseed. The final value is additional input to the second generate call.

114 The following paragraphs contain more information on some of the input values to be generated/selected by the evaluator.

**Entropy input:** the length of the entropy input value must equal the seed length.

**Nonce:** If a nonce is supported (CTR_DRBG with no Derivation Function does not use a nonce), the nonce bit length is one-half the seed length.

**Personalization string:** The length of the personalization string must be <= seed length. If the implementation only supports one personalization string length, then the same length can be used for both values. If more than one string length is support, the evaluator shall use personalization strings of two different lengths. If the implementation does not use a personalization string, no value needs to be supplied.

**Additional input:** the additional input bit lengths have the same defaults and restrictions as the personalization string lengths.

### 2.2.9 FCS_HTTPS_EXT.1 HTTPS Protocol

#### 2.2.9.1 Tests

115 The evaluator shall perform the following tests:

a) Test 1: The evaluator shall attempt to establish an HTTPS connection with a web server, observe the traffic with a packet analyzer, and verify that the connection succeeds and that the traffic is identified as TLS or HTTPS.

116 Other tests are performed in conjunction with the TLS evaluation activities.

117 Certificate validity shall be tested in accordance with testing performed for FIA_X509_EXT.1, and the evaluator shall perform the following test:

b) Test 2: The evaluator shall demonstrate that using a certificate without a valid certification path results in an application notification. Using the administrative guidance, the evaluator shall then load a valid certificate and certification path, and demonstrate that the function succeeds. The evaluator then shall delete one of the certificates, and show that the selection listed in the ST occurs.

### 2.2.10 FCS_IPSEC_EXT.1 IPsec Protocol

#### 2.2.10.1 TSS

**FCS_IPSEC_EXT.1.1**

118 The evaluator shall examine the TSS and determine that it describes what takes place when a packet is processed by the TOE, e.g., the algorithm used to process the packet. The TSS describes how the SPD is implemented and the rules for processing both inbound and outbound packets in terms of the IPsec policy. The TSS describes the rules that are available and the resulting actions available after matching a rule. The TSS describes how those rules and actions form the SPD in terms of the BYPASS (e.g., no encryption),

DISCARD (e.g., drop the packet), and PROTECT (e.g., encrypt the packet) actions defined in RFC 4301.

119 As noted in section 4.4.1 of RFC 4301, the processing of entries in the SPD is non-trivial and the evaluator shall determine that the description in the TSS is sufficient to determine which rules will be applied given the rule structure implemented by the TOE. For example, if the TOE allows specification of ranges, conditional rules, etc., the evaluator shall determine that the description of rule processing (for both inbound and outbound packets) is sufficient to determine the action that will be applied, especially in the case where two different rules may apply. This description shall cover both the initial packets (that is, no SA is established on the interface or for that particular packet) as well as packets that are part of an established SA.

**FCS_IPSEC_EXT.1.3**

120 The evaluator checks the TSS to ensure it states that the VPN can be established to operate in transport mode and/or tunnel mode (as identified in FCS_IPSEC_EXT.1.3).

**FCS_IPSEC_EXT.1.4**

121 The evaluator shall examine the TSS to verify that the algorithms AES-CBC-128 and AES-CBC-256 are implemented. If the ST author has selected either AES-GCM-128 or AES-GCM-256 in the requirement, then the evaluator verifies the TSS describes these as well. In addition, the evaluator ensures that the SHA-based HMAC algorithm conforms to the algorithms specified in FCS_COP.1(4) Cryptographic Operations (for keyed-hash message authentication).

**FCS_IPSEC_EXT.1.5**

122 The evaluator shall examine the TSS to verify that IKEv1 and/or IKEv2 are implemented.

123 For IKEv1 implementations, the evaluator shall examine the TSS to ensure that, in the description of the IPsec protocol, it states that aggressive mode is not used for IKEv1 Phase 1 exchanges, and that only main mode is used. It may be that this is a configurable option.

**FCS_IPSEC_EXT.1.6**

124 The evaluator shall ensure the TSS identifies the algorithms used for encrypting the IKEv1 and/or IKEv2 payload, and that the algorithms AES-CBC-128, AES-CBC-256 are specified, and if others are chosen in the selection of the requirement, those are included in the TSS discussion.

**FCS_IPSEC_EXT.1.7**

125 The evaluator shall ensure the TSS identifies the lifetime configuration method used for limiting the IKEv1 Phase 1 SA lifetime and/or the IKEv2 SA lifetime. The evaluator shall verify that the selection made here corresponds to the selection in FCS_IPSEC_EXT.1.5.

**FCS_IPSEC_EXT.1.8**

126 The evaluator shall ensure the TSS identifies the lifetime configuration method used for limiting the IKEv1 Phase 2 SA lifetime and/or the IKEv2 Child SA lifetime. The evaluator shall verify that the selection made here corresponds to the selection in FCS_IPSEC_EXT.1.5.

**FCS_IPSEC_EXT.1.9**

127 The evaluator shall check to ensure that, for each DH group supported, the TSS describes the process for generating "x". The evaluator shall verify that the TSS indicates that the random number generated that meets the requirements in this PP is used, and that the length of "x" meets the stipulations in the requirement.

**FCS_IPSEC_EXT.1.11**

128 The evaluator shall check to ensure that the DH groups specified in the requirement are listed as being supported in the TSS. If there is more than one DH group supported, the evaluator checks to ensure the TSS describes how a particular DH group is specified/negotiated with a peer.

**FCS_IPSEC_EXT.1.12**

129 The evaluator shall check that the TSS describes the potential strengths (in terms of the number of bits in the symmetric key) of the algorithms that are allowed for the IKE and ESP exchanges. The TSS shall also describe the checks that are done when negotiating IKEv1 Phase 2 and/or IKEv2 CHILD_SA suites to ensure that the strength (in terms of the number of bits of key in the symmetric algorithm) of the negotiated algorithm is less than or equal to that of the IKE SA this is protecting the negotiation.

**FCS_IPSEC_EXT.1.13**

130 The evaluator ensures that the TSS identifies RSA and/or ECDSA as being used to perform peer authentication. The description must be consistent with the algorithms as specified in FCS_COP.1(2) Cryptographic Operations (for cryptographic signature).

131 If pre-shared keys are chosen in the selection, the evaluator shall check to ensure that the TSS describes how pre-shared keys are established and used in authentication of IPsec connections. The description in the TSS shall also indicate how pre-shared key establishment is accomplished for TOEs that can generate a pre-shared key as well as TOEs that simply use a pre-shared key.

**FCS_IPSEC_EXT.1.14**

132 The evaluator shall verify that the TSS describes how the DN in the certificate is compared to the expected DN.

#### 2.2.10.2 Guidance Documentation

**FCS_IPSEC_EXT.1.1**

133 The evaluator shall examine the guidance documentation to verify it instructs the Administrator how to construct entries into the SPD that specify a rule for processing a packet. The description includes all three cases – a rule that ensures packets are encrypted/decrypted, dropped, and flow through the TOE without being encrypted. The evaluator shall determine that the description in the guidance documentation is consistent with the description in the TSS, and that the level of detail in the guidance documentation is sufficient to allow the administrator to set up the SPD in an unambiguous fashion. This includes a discussion of how ordering of rules impacts the processing of an IP packet.

**FCS_IPSEC_EXT.1.3**

134 The evaluator shall confirm that the guidance documentation contains instructions on how to configure the connection in each mode selected.

**FCS_IPSEC_EXT.1.4**

135 The evaluator checks the guidance documentation to ensure it provides instructions on how to configure the TOE to use the algorithms, and if either AES-GCM-128 or AES-GCM-256 have been selected the guidance instructs how to use these as well.

**FCS_IPSEC_EXT.1.5**

136 The evaluator shall check the guidance documentation to ensure it instructs the administrator how to configure the TOE to use IKEv1 and/or IKEv2 (as selected), and uses the guidance to configure the TOE to perform NAT traversal for the following test (if selected).

137 If the IKEv1 Phase 1 mode requires configuration of the TOE prior to its operation, the evaluator shall check the guidance documentation to ensure that instructions for this configuration are contained within that guidance.

**FCS_IPSEC_EXT.1.6**

138 The evaluator ensures that the guidance documentation describes the configuration of the mandated algorithms, as well as any additional algorithms selected in the requirement. The guidance is then used to configure the TOE to perform the following test for each ciphersuite selected.

**FCS_IPSEC_EXT.1.7**

139 The evaluator shall verify that the values for SA lifetimes can be configured and that the instructions for doing so are located in the guidance documentation. If time-based limits are supported, the evaluator ensures that the Administrator is able to configure Phase 1 SA values for 24 hours. Currently there are no values mandated for the number of bytes, the

evaluator just ensures that this can be configured if selected in the requirement.

**FCS_IPSEC_EXT.1.8**

140 The evaluator shall verify that the values for SA lifetimes can be configured and that the instructions for doing so are located in the guidance documentation. If time-based limits are supported, the evaluator ensures that the Administrator is able to configure Phase 2 SA values for 8 hours. Currently there are no values mandated for the number of bytes, the evaluator just ensures that this can be configured if selected in the requirement.

**FCS_IPSEC_EXT.1.11**

141 The evaluator ensures that the guidance documentation describes the configuration of the mandated algorithms, as well as any additional algorithms selected in the requirement. The guidance is then used to configure the TOE to perform the following test for each ciphersuite selected.

**FCS_IPSEC_EXT.1.13**

142 The evaluator ensures the guidance documentation describes how to set up the TOE to use certificates with RSA and/or ECDSA signatures and public keys.

143 The evaluator shall check that the guidance documentation describes how pre-shared keys are to be generated and established. The description in the guidance documentation shall also indicate how pre-shared key establishment is accomplished for TOEs that can generate a pre-shared key as well as TOEs that simply use a pre-shared key.

144 In order to construct the environment and configure the TOE for the following tests, the evaluator will ensure that the guidance documentation describes how to configure the TOE to connect to a trusted CA, and ensure a valid certificate for that CA is loaded into the TOE and marked "trusted".

**FCS_IPSEC_EXT.1.14**

145 The evaluator shall ensure that the guidance documentation includes configuration of the expected DN for the connection.

#### 2.2.10.3 Tests

**FCS_IPSEC_EXT.1.1**

146 The evaluator uses the guidance documentation to configure the TOE to carry out the following tests:

a) Test 1: The evaluator shall configure the SPD such that there is a rule for dropping a packet, encrypting a packet, and allowing a packet to flow in plaintext. The selectors used in the construction of the rule shall be different such that the evaluator can generate a packet and

send packets to the gateway with the appropriate fields (fields that are used by the rule - e.g., the IP addresses, TCP/UDP ports) in the packet header. The evaluator performs both positive and negative test cases for each type of rule (e.g. a packet that matches the rule and another that does not match the rule). The evaluator observes via the audit trail, and packet captures that the TOE exhibited the expected behavior: appropriate packets were dropped, allowed to flow without modification, encrypted by the IPsec implementation.

b) Test 2: The evaluator shall devise several tests that cover a variety of scenarios for packet processing. As with Test 1, the evaluator ensures both positive and negative test cases are constructed. These scenarios must exercise the range of possibilities for SPD entries and processing modes as outlined in the TSS and guidance documentation. Potential areas to cover include rules with overlapping ranges and conflicting entries, inbound and outbound packets, and packets that establish SAs as well as packets that belong to established SAs. The evaluator shall verify, via the audit trail and packet captures, for each scenario that the expected behavior is exhibited, and is consistent with both the TSS and the guidance documentation.

**FCS_IPSEC_EXT.1.2**

147 The assurance activity for this element is performed in conjunction with the activities for FCS_IPSEC_EXT.1.1.

148 The evaluator uses the guidance documentation to configure the TOE to carry out the following tests:

149 The evaluator shall configure the SPD such that there is a rule for dropping a packet, encrypting a packet, and allowing a packet to flow in plaintext. The evaluator may use the SPD that was created for verification of FCS_IPSEC_EXT.1.1. The evaluator shall construct a network packet that matches the rule to allow the packet to flow in plaintext and send that packet. The evaluator should observe that the network packet is passed to the proper destination interface with no modification. The evaluator shall then modify a field in the packet header; such that it no longer matches the evaluator-created entries (there may be a "TOE created" final entry that discards packets that do not match any previous entries). The evaluator sends the packet, and observes that the packet was dropped.

**FCS_IPSEC_EXT.1.3**

150 The evaluator shall perform the following test(s) based on the selections chosen:

a) Test 1 (conditional): If tunnel mode is selected, the evaluator uses the guidance documentation to configure the TOE to operate in tunnel mode and also configures a VPN peer to operate in tunnel mode. The evaluator configures the TOE and the VPN peer to use any of the allowable cryptographic algorithms, authentication methods, etc. to

ensure an allowable SA can be negotiated. The evaluator shall then initiate a connection from the TOE to connect to the VPN peer. The evaluator observes (for example, in the audit trail and the captured packets) that a successful connection was established using the tunnel mode.

b) Test 2: The evaluator uses the guidance documentation to configure the TOE to operate in transport mode and also configures a VPN peer to operate in transport mode. The evaluator configures the TOE and the VPN peer to use any of the allowed cryptographic algorithms, authentication methods, etc. to ensure an allowable SA can be negotiated. The evaluator then initiates a connection from the TOE to connect to the VPN peer. The evaluator observes (for example, in the audit trail and the captured packets) that a successful connection was established using the transport mode.

**FCS_IPSEC_EXT.1.4**

151 The evaluator shall configure the TOE as indicated in the guidance documentation configuring the TOE to use each of the supported algorithms, attempt to establish a connection using ESP, and verify that the attempt succeeds.

**FCS_IPSEC_EXT.1.5**

152 Tests are performed in conjunction with the other IPsec evaluation activities.

153 (conditional): The evaluator shall configure the TOE as indicated in the guidance documentation, and attempt to establish a connection using an IKEv1 Phase 1 connection in aggressive mode. This attempt should fail. The evaluator should then show that main mode exchanges are supported.

154 (conditional): The evaluator shall configure the TOE so that it will perform NAT traversal processing as described in the TSS and RFC 5996, section 2.23. The evaluator shall initiate an IPsec connection and determine that the NAT is successfully traversed.

**FCS_IPSEC_EXT.1.6**

155 The evaluator shall configure the TOE to use the ciphersuite under test to encrypt the IKEv1 and/or IKEv2 payload and establish a connection with a peer device, which is configured to only accept the payload encrypted using the indicated ciphersuite. The evaluator will confirm the algorithm was that used in the negotiation.

**FCS_IPSEC_EXT.1.7**

156 When testing this functionality, the evaluator needs to ensure that both sides are configured appropriately. From the RFC "A difference between IKEv1 and IKEv2 is that in IKEv1 SA lifetimes were negotiated. In IKEv2, each end of the SA is responsible for enforcing its own lifetime policy on the SA and rekeying the SA when necessary. If the two ends have different lifetime policies, the end with the shorter lifetime will end up always being the one to

request the rekeying. If the two ends have the same lifetime policies, it is possible that both will initiate a rekeying at the same time (which will result in redundant SAs). To reduce the probability of this happening, the timing of rekeying requests SHOULD be jittered."

157 Each of the following tests shall be performed for each version of IKE selected in the FCS_IPSEC_EXT.1.5 protocol selection:

a) Test 1 (Conditional): The evaluator shall configure a maximum lifetime in terms of the number of bytes allowed following the guidance documentation. The evaluator shall configure a test peer with a byte lifetime that exceeds the lifetime of the TOE. The evaluator shall establish an SA between the TOE and the test peer, and determine that once the allowed number of bytes through this SA is exceeded, a new SA is negotiated. The evaluator shall verify that the TOE initiates a Phase 1 negotiation.

b) Test 2 (Conditional): The evaluator shall configure a maximum lifetime of 24 hours for the Phase 1 SA following the guidance documentation. The evaluator shall configure a test peer with a lifetime that exceeds the lifetime of the TOE. The evaluator shall establish an SA between the TOE and the test peer, maintain the Phase 1 SA for 24 hours, and determine that once 24 hours has elapsed, a new Phase 1 SA is negotiated. The evaluator shall verify that the TOE initiates a Phase 1 negotiation.

**FCS_IPSEC_EXT.1.8**

158 When testing this functionality, the evaluator needs to ensure that both sides are configured appropriately. From the RFC "A difference between IKEv1 and IKEv2 is that in IKEv1 SA lifetimes were negotiated. In IKEv2, each end of the SA is responsible for enforcing its own lifetime policy on the SA and rekeying the SA when necessary. If the two ends have different lifetime policies, the end with the shorter lifetime will end up always being the one to request the rekeying. If the two ends have the same lifetime policies, it is possible that both will initiate a rekeying at the same time (which will result in redundant SAs). To reduce the probability of this happening, the timing of rekeying requests SHOULD be jittered."

159 Each of the following tests shall be performed for each version of IKE selected in the FCS_IPSEC_EXT.1.5 protocol selection:

a) Test 1 (Conditional): The evaluator shall configure a maximum lifetime in terms of the number of bytes allowed following the guidance documentation. The evaluator shall configure a test peer with a byte lifetime that exceeds the lifetime of the TOE. The evaluator shall establish an SA between the TOE and the test peer, and determine that once the allowed number of bytes through this SA is exceeded, a new SA is negotiated. The evaluator shall verify that the TOE initiates a Phase 2 negotiation.

b) Test 2 (Conditional): The evaluator shall configure a maximum lifetime of 8 hours for the Phase 2 SA following the guidance documentation. The evaluator shall configure a test peer with a lifetime that exceeds the lifetime of the TOE. The evaluator shall establish an SA between the TOE and the test peer, maintain the Phase 1 SA for 8 hours, and determine that once 8 hours has elapsed, a new Phase 2 SA is negotiated. The evaluator shall verify that the TOE initiates a Phase 2 negotiation.

**FCS_IPSEC_EXT.1.10**

160 (conditional) If the first selection is chosen, the evaluator shall check to ensure that, for each DH group supported, the TSS describes the process for generating each nonce. The evaluator shall verify that the TSS indicates that the random number generated that meets the requirements in this PP is used, and that the length of the nonces meet the stipulations in the requirement.

161 (conditional) If the second selection is chosen, the evaluator shall check to ensure that, for each PRF hash supported, the TSS describes the process for generating each nonce. The evaluator shall verify that the TSS indicates that the random number generated that meets the requirements in this PP is used, and that the length of the nonces meet the stipulations in the requirement.

**FCS_IPSEC_EXT.1.11**

162 For each supported DH group, the evaluator shall test to ensure that all supported IKE protocols can be successfully completed using that particular DH group.

**FCS_IPSEC_EXT.1.12**

163 The evaluator simply follows the guidance to configure the TOE to perform the following tests.

a) Test 1: This test shall be performed for each version of IKE supported. The evaluator shall successfully negotiate an IPsec connection using each of the supported algorithms and hash functions identified in the requirements.

b) Test 2: This test shall be performed for each version of IKE supported. The evaluator shall attempt to establish an SA for ESP that selects an encryption algorithm with more strength than that being used for the IKE SA (i.e., symmetric algorithm with a key size larger than that being used for the IKE SA). Such attempts should fail.

c) Test 3: This test shall be performed for each version of IKE supported. The evaluator shall attempt to establish an IKE SA using an algorithm that is not one of the supported algorithms and hash functions identified in the requirements. Such an attempt should fail.

d) Test 4: This test shall be performed for each version of IKE supported. The evaluator shall attempt to establish an SA for ESP (assumes the proper parameters where used to establish the IKE SA)

that selects an encryption algorithm that is not identified in FCS_IPSEC_EXT.1.4. Such an attempt should fail.

**FCS_IPSEC_EXT.1.13**

164 For efficiency sake, the testing that is performed may be combined with the testing for FIA_X509_EXT.1, FIA_X509_EXT.2 (for IPsec connections), and FCS_IPSEC_EXT.1.1. The following tests shall be repeated for each peer authentication selected in the FCS_IPSEC_EXT.1.1 selection above:

a) Test 1: The evaluator shall configure the TOE to use a private key and associated certificate signed by a trusted CA and shall establish an IPsec connection with the peer.

b) Test 2 [conditional]: The evaluator shall generate a pre-shared key off-TOE and use it, as indicated in the guidance documentation, to establish an IPsec connection with the peer.

**FCS_IPSEC_EXT.1.14**

165 The evaluator shall, if necessary, configure the expected DN according to the guidance documentation. The evaluator shall send a peer certificate signed by a trusted CA with a DN that does not match an expected DN and verify that the TOE denies the connection.

### 2.2.11 FCS_SSHC_EXT.1 SSH Client

#### 2.2.11.1 TSS

**FCS_SSHC_EXT.1.2**

166 The evaluator shall check to ensure that the TSS contains a description of the public key algorithms that are acceptable for use for authentication, that this list conforms to FCS_SSHC_EXT.1.5, and ensure that password-based authentication methods are also allowed.

**FCS_SSHC_EXT.1.3**

167 The evaluator shall check that the TSS describes how “large packets” in terms of RFC 4253 are detected and handled.

**FCS_SSHC_EXT.1.4**

168 The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that optional characteristics are specified, and the encryption algorithms supported are specified as well. The evaluator shall check the TSS to ensure that the encryption algorithms specified are identical to those listed for this component.

**FCS_SSHC_EXT.1.5**

169 The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that optional characteristics are specified, and the public key algorithms supported are specified as well. The evaluator shall check the TSS to ensure that the public key algorithms specified are identical to those listed for this component.

**FCS_SSHC_EXT.1.6**

170 The evaluator shall check the TSS to ensure that it lists the supported data integrity algorithms, and that that list corresponds to the list in this component.

**FCS_SSHC_EXT.1.7**

171 The evaluator shall check the TSS to ensure that it lists the supported key exchange algorithms, and that that list corresponds to the list in this component.

#### 2.2.11.2 Guidance Documentation

**FCS_SSHC_EXT.1.4**

172 The evaluator shall also check the guidance documentation to ensure that it contains instructions on configuring the TOE so that SSH conforms to the description in the TSS (for instance, the set of algorithms advertised by the TOE may have to be restricted to meet the requirements).

**FCS_SSHC_EXT.1.5**

173 The evaluator shall also check the guidance documentation to ensure that it contains instructions on configuring the TOE so that SSH conforms to the description in the TSS (for instance, the set of algorithms advertised by the TOE may have to be restricted to meet the requirements).

**FCS_SSHC_EXT.1.6**

174 The evaluator shall also check the guidance documentation to ensure that it contains instructions to the administrator on how to ensure that only the allowed data integrity algorithms are used in SSH connections with the TOE (specifically, that the "none" MAC algorithm is not allowed).

**FCS_SSHC_EXT.1.7**

175 The evaluator shall also check the guidance documentation to ensure that it contains instructions to the administrator on how to ensure that only the allowed key exchange algorithms are used in SSH connections with the TOE.

#### 2.2.11.3 Tests

**FCS_SSHC_EXT.1.2**

176 Test 1: The evaluator shall, for each public key algorithm supported, show that the TOE supports the use of that public key algorithm to authenticate a

user connection to an SSH server. Any configuration activities required to support this test shall be performed according to instructions in the guidance documentation.

177 Test 2: Using the guidance documentation, the evaluator shall configure the TOE to perform password-based authentication to an SSH server, and demonstrate that a user can be successfully authenticated by the TOE to an SSH server using a password as an authenticator.

**FCS_SSHC_EXT.1.3**

178 The evaluator shall demonstrate that if the TOE receives a packet larger than that specified in this component, that packet is dropped.

**FCS_SSHC_EXT.1.4**

179 Test 1: The evaluator shall establish a SSH connection using each of the encryption algorithms specified by the requirement. It is sufficient to observe (on the wire) the successful negotiation of the algorithm to satisfy the intent of the test.

180 Test 2: The evaluator shall configure an SSH server to only allow the 3des-cbc encryption algorithm and no other encryption algorithms. The evaluator shall attempt to establish an SSH connection from the TOE to the SSH server and observe that the connection is rejected.

**FCS_SSHC_EXT.1.5**

181 Test 1: The evaluator shall establish a SSH connection using each of the public key algorithms specified by the requirement to authenticate an SSH server to the TOE. It is sufficient to observe (on the wire) the successful negotiation of the algorithm to satisfy the intent of the test.

182 Test 2: The evaluator shall configure an SSH server to only allow the ssh-dsa public key algorithm and no other public key algorithms. The evaluator shall attempt to establish an SSH connection from the TOE to the SSH server and observe that the connection is rejected.

**FCS_SSHC_EXT.1.6**

183 Test 1: The evaluator shall establish a SSH connection using each of the integrity algorithms specified by the requirement. It is sufficient to observe (on the wire) the successful negotiation of the algorithm to satisfy the intent of the test.

184 Test 2: The evaluator shall configure an SSH server to only allow the “none” MAC algorithm. The evaluator shall attempt to connect from the TOE to the SSH server and observe that the attempt fails.

185 Test 3: The evaluator shall configure an SSH server to only allow the hmac-md5 MAC algorithm. The evaluator shall attempt to connect from the TOE to the SSH server and observe that the attempt fails.

**FCS_SSHC_EXT.1.7**

186 Test 1: The evaluator shall configure an SSH server to permit all allowed key exchange methods. The evaluator shall attempt to connect from the TOE to the SSH server using each allowed key exchange method, and observe that each attempt succeeds.

**FCS_SSHC_EXT.1.8**

187 The evaluator shall configure the TOE to create a log entry when a rekey occurs. The evaluator shall connect to the TOE with an SSH client and cause 2^28 packets to be transmitted from the client to the TOE, and subsequently review the audit log to ensure that a rekey occurred.

**FCS_SSHC_EXT.1.9**

188 Test 1: The evaluator shall delete all entries in the TOE's list of recognized SSH server host keys and, if selected, all entries in the TOE's list of trusted certification authorities. The evaluator shall initiate a connection from the TOE to an SSH server. The evaluator shall ensure that the TOE either rejects the connection or displays the SSH server's public key (either the key bytes themselves or a hash of the key using any allowed hash algorithm) and prompts the user to accept or deny the key before continuing the connection.

189 Test 2: The evaluator shall add an entry associating a host name with a public key into the TOE's local database. The evaluator shall replace, on the corresponding SSH server, the server's host key with a different host key. The evaluator shall initiate a connection from the TOE to the SSH server using password-based authentication, shall ensure that the TOE rejects the connection, and shall ensure that the password was not transmitted to the SSH server (for example, by instrumenting the SSH server with a debugging capability to output received passwords).

### 2.2.12 FCS_SSHS_EXT.1 SSH Server

#### 2.2.12.1 TSS

**FCS_SSHS_EXT.1.2**

190 The evaluator shall check to ensure that the TSS contains a description of the public key algorithms that are acceptable for use for authentication, that this list conforms to FCS_SSHS_EXT.1.5, and ensure that password-based authentication methods are also allowed.

**FCS_SSHS_EXT.1.3**

191 The evaluator shall check that the TSS describes how "large packets" in terms of RFC 4253 are detected and handled.

**FCS_SSHS_EXT.1.4**

192 The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that optional characteristics are specified, and

the encryption algorithms supported are specified as well. The evaluator shall check the TSS to ensure that the encryption algorithms specified are identical to those listed for this component.

**FCS_SSHS_EXT.1.5**

193 The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that optional characteristics are specified, and the public key algorithms supported are specified as well. The evaluator shall check the TSS to ensure that the public key algorithms specified are identical to those listed for this component.

**FCS_SSHS_EXT.1.6**

194 The evaluator shall check the TSS to ensure that it lists the supported data integrity algorithms, and that that list corresponds to the list in this component.

**FCS_SSHS_EXT.1.7**

195 The evaluator shall check the TSS to ensure that it lists the supported key exchange algorithms, and that that list corresponds to the list in this component.

#### 2.2.12.2 Guidance Documentation

**FCS_SSHS_EXT.1.4**

196 The evaluator shall also check the guidance documentation to ensure that it contains instructions on configuring the TOE so that SSH conforms to the description in the TSS (for instance, the set of algorithms advertised by the TOE may have to be restricted to meet the requirements).

**FCS_SSHS_EXT.1.5**

197 The evaluator shall also check the guidance documentation to ensure that it contains instructions on configuring the TOE so that SSH conforms to the description in the TSS (for instance, the set of algorithms advertised by the TOE may have to be restricted to meet the requirements).

**FCS_SSHS_EXT.1.6**

198 The evaluator shall also check the guidance documentation to ensure that it contains instructions to the administrator on how to ensure that only the allowed data integrity algorithms are used in SSH connections with the TOE (specifically, that the “none” MAC algorithm is not allowed).

**FCS_SSHS_EXT.1.7**

199 The evaluator shall also check the guidance documentation to ensure that it contains instructions to the administrator on how to ensure that only the allowed key exchange algorithms are used in SSH connections with the TOE.

#### 2.2.12.3 Tests

**FCS_SSHS_EXT.1.2**

200 Test 1: The evaluator shall, for each public key algorithm supported, show that the TOE supports the use of that public key algorithm to authenticate a user connection. Any configuration activities required to support this test shall be performed according to instructions in the guidance documentation.

201 Test 2: The evaluator shall choose one public key algorithm supported by the TOE. The evaluator shall generate a new key pair for that algorithm without configuring the TOE to recognize the public key for authentication. The evaluator shall use an SSH client to attempt to connect to the TOE with the new key pair and demonstrate that authentication fails.

202 Test 3: Using the guidance documentation, the evaluator shall configure the TOE to accept password-based authentication, and demonstrate that a user can be successfully authenticated to the TOE over SSH using a password as an authenticator.

203 Test 4: The evaluator shall use an SSH client, enter an incorrect password to attempt to authenticate to the TOE, and demonstrate that the authentication fails.

**FCS_SSHS_EXT.1.3**

204 The evaluator shall demonstrate that if the TOE receives a packet larger than that specified in this component, that packet is dropped.

**FCS_SSHS_EXT.1.4**

205 Test 1: The evaluator shall establish a SSH connection using each of the encryption algorithms specified by the requirement. It is sufficient to observe (on the wire) the successful negotiation of the algorithm to satisfy the intent of the test.

206 Test 2: The evaluator shall configure an SSH client to only allow the 3des-cbc encryption algorithm and no other encryption algorithms. The evaluator shall attempt to establish an SSH connection from the SSH client to the TOE and observe that the connection is rejected.

**FCS_SSHS_EXT.1.5**

207 Test 1: The evaluator shall establish a SSH connection using each of the public key algorithms specified by the requirement to authenticate the TOE to an SSH client. It is sufficient to observe (on the wire) the successful negotiation of the algorithm to satisfy the intent of the test.

208 Test 2: The evaluator shall configure an SSH client to only allow the ssh-dsa public key algorithm and no other public key algorithms. The evaluator shall attempt to establish an SSH connection from the SSH client to the TOE and observe that the connection is rejected.

**FCS_SSHS_EXT.1.6**

209 Test 1: The evaluator shall establish a SSH connection using each of the integrity algorithms specified by the requirement. It is sufficient to observe (on the wire) the successful negotiation of the algorithm to satisfy the intent of the test.

210 Test 2: The evaluator shall configure an SSH client to only allow the "none" MAC algorithm. The evaluator shall attempt to connect from the SSH client to the TOE and observe that the attempt fails.

211 Test 3: The evaluator shall configure an SSH client to only allow the hmac-md5 MAC algorithm. The evaluator shall attempt to connect from the SSH client to the TOE and observe that the attempt fails.

**FCS_SSHS_EXT.1.7**

212 Test 1: The evaluator shall configure an SSH client to only allow the diffie-hellman-group1-sha1 key exchange. The evaluator shall attempt to connect from the SSH client to the TOE and observe that the attempt fails.

213 Test 2: For each allowed key exchange method, the evaluator shall configure an SSH client to only allow that method for key exchange, attempt to connect from the client to the TOE, and observe that the attempt succeeds.

**FCS_SSHS_EXT.1.8**

214 The evaluator shall configure the TOE to create a log entry when a rekey occurs. The evaluator shall connect to the TOE with an SSH client and cause 2^28 packets to be transmitted from the client to the TOE, and subsequently review the audit log to ensure that a rekey occurred.

### 2.2.13 FCS_TLSC_EXT.1 Extended: TLS Client Protocol

#### 2.2.13.1 TSS

**FCS_TLSC_EXT.1.1**

215 The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that the ciphersuites supported are specified. The evaluator shall check the TSS to ensure that the ciphersuites specified include those listed for this component.

**FCS_TLSC_EXT.1.2**

216 The evaluator shall ensure that the TSS describes the client's method of establishing all reference identifiers from the administrator/application-configured reference identifier, including which types of reference identifiers are supported (e.g Common Name, DNS Name, URI Name, Service Name, or other application-specific Subject Alternative Names) and whether IP addresses and wildcards are supported. The evaluator shall ensure that this description identifies whether and the manner in which certificate pinning is supported or used by the TOE.

**FCS_TLSC_EXT.1.4**

217 The evaluator shall verify that TSS describes the Supported Elliptic Curves Extension and whether the required behaviour is performed by default or may be configured.

#### 2.2.13.2 Guidance Documentation

**FCS_TLSC_EXT.1.1**

218 The evaluator shall also check the guidance documentation to ensure that it contains instructions on configuring the TOE so that TLS conforms to the description in the TSS.

**FCS_TLSC_EXT.1.2**

219 The evaluator shall verify that the AGD guidance includes instructions for setting the reference identifier to be used for the purposes of certificate validation in TLS.

**FCS_TLSC_EXT.1.4**

220 If the TSS indicates that the Supported Elliptic Curves Extension must be configured to meet the requirement, the evaluator shall verify that AGD guidance includes configuration of the Supported Elliptic Curves Extension.

#### 2.2.13.3 Tests

**FCS_TLSC_EXT.1.1**

221 Test 1: The evaluator shall establish a TLS connection using each of the ciphersuites specified by the requirement. This connection may be established as part of the establishment of a higher-level protocol, e.g., as part of an HTTPS session. It is sufficient to observe the successful negotiation of a ciphersuite to satisfy the intent of the test; it is not necessary to examine the characteristics of the encrypted traffic in an attempt to discern the ciphersuite being used (for example, that the cryptographic algorithm is 128-bit AES and not 256-bit AES).

222 Test 2: The evaluator shall attempt to establish the connection using a server with a server certificate that contains the Server Authentication purpose in the extendedKeyUsage field and verify that a connection is established. The evaluator will then verify that the client rejects an otherwise valid server certificate that lacks the Server Authentication purpose in the extendedKeyUsage field and a connection is not established. Ideally, the two certificates should be identical except for the extendedKeyUsage field.

223 Test 3: The evaluator shall send a server certificate in the TLS connection that the does not match the server-selected ciphersuite (for example, send a ECDSA certificate while using the TLS_RSA_WITH_AES_128_CBC_SHA ciphersuite). The evaluator shall verify that the TOE disconnects after receiving the server's Certificate handshake message.

224 Test 4: The evaluator shall configure the server to select the TLS_NULL_WITH_NULL_NULL ciphersuite and verify that the client denies the connection. Test 2 in FCS_TLSS_EXT.1.1 or FCS_TLSS_EXT.2.1 can be used as a substitute for this test.

225 Test 5: The evaluator perform the following modifications to the traffic:

a) Change the TLS version selected by the server in the Server Hello to a non-supported TLS version (for example 1.3 represented by the two bytes 03 04) and verify that the client rejects the connection.

b) Modify at least one byte in the server's nonce in the Server Hello handshake message, and verify that the client rejects the Server Key Exchange handshake message (if using a DHE or ECDHE ciphersuite) or that the server denies the client's Finished handshake message.

c) Modify the server's selected ciphersuite in the Server Hello handshake message to be a ciphersuite not presented in the Client Hello handshake message. The evaluator shall verify that the client rejects the connection after receiving the Server Hello.

d) Modify the signature block in the Server's Key Exchange handshake message, and verify that the client rejects the connection after receiving the Server Key Exchange message.

e) Modify a byte in the Server Finished handshake message, and verify that the client sends a fatal alert upon receipt and does not send any application data.

f) Send a garbled message from the Server after the Server has issued the ChangeCipherSpec message and verify that the client denies the connection.

**FCS_TLSC_EXT.1.2**

226 The evaluator shall configure the reference identifier according to the AGD guidance and perform the following tests during a TLS connection:

a) Test 1: The evaluator shall present a server certificate that does not contain an identifier in either the Subject Alternative Name (SAN) or Common Name (CN) that matches the reference identifier. The evaluator shall verify that the connection fails.

b) Test 2: The evaluator shall present a server certificate that contains a CN that matches the reference identifier, contains the SAN extension, but does not contain an identifier in the SAN that matches the reference identifier. The evaluator shall verify that the connection fails. The evaluator shall repeat this test for each supported SAN type.

c) Test 3: The evaluator shall present a server certificate that contains a CN that matches the reference identifier and does not contains the SAN extension. The evaluator shall verify that the connection succeeds.

d) Test 4: The evaluator shall present a server certificate that contains a CN that does not match the reference identifier but does contain an identifier in the SAN that matches. The evaluator shall verify that the connection succeeds.

e) Test 5: The evaluator shall perform the following wildcard tests with each supported type of reference identifier:

1) The evaluator shall present a server certificate containing a wildcard that is not in the left-most label of the presented identifier (e.g. foo.*.example.com) and verify that the connection fails.

2) The evaluator shall present a server certificate containing a wildcard in the left-most label (e.g. *.example.com). The evaluator shall configure the reference identifier with a single left-most label (e.g. foo.example.com) and verify that the connection succeeds. The evaluator shall configure the reference identifier without a left-most label as in the certificate (e.g. example.com) and verify that the connection fails. The evaluator shall configure the reference identifier with two left-most labels (e.g. bar.foo.example.come) and verify that the connection fails.

f) Test 6: [conditional] If URI or Service name reference identifiers are supported, the evaluator shall configure the DNS name and the service identifier. The evaluator shall present a server certificate containing the correct DNS name and service identifier in the URIName or SRVName fields of the SAN and verify that the connection succeeds. The evaluator shall repeat this test with the wrong service identifier (but correct DNS name) and verify that the connection fails.

g) Test 7: [conditional] If pinned certificates are supported the evaluator shall present a certificate that does not match the pinned certificate and verify that the connection fails.

**FCS_TLSC_EXT.1.3**

227 Test 1: The evaluator shall demonstrate that using a certificate without a valid certification path results in the function failing. Using the administrative guidance, the evaluator shall then load a certificate or certificates needed to validate the certificate to be used in the function, and demonstrate that the function succeeds. If the certificate is validated and a trusted channel is established, the test passes. The evaluator then shall delete one of the certificates, and show that the certificate is not validated and the trusted channel is not established..

**FCS_TLSC_EXT.1.4**

228 Test 1: The evaluator shall configure the server to perform an ECDHE key exchange in the TLS connection using a non-supported curve (for example P-192) and shall verify that the TOE disconnects after receiving the server's Key Exchange handshake message.

### 2.2.14 FCS_TLSC_EXT.2 Extended: TLS Client Protocol with authentication

#### 2.2.14.1 TSS

**FCS_TLSC_EXT.2.1**

229 The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that the ciphersuites supported are specified. The evaluator shall check the TSS to ensure that the ciphersuites specified include those listed for this component.

**FCS_TLSC_EXT.2.2**

230 The evaluator shall ensure that the TSS describes the client's method of establishing all reference identifiers from the administrator/application-configured reference identifier, including which types of reference identifiers are supported (e.g Common Name, DNS Name, URI Name, Service Name, or other application-specific Subject Alternative Names) and whether IP addresses and wildcards are supported. The evaluator shall ensure that this description identifies whether and the manner in which certificate pinning is supported or used by the TOE.

**FCS_TLSC_EXT.2.4**

231 The evaluator shall verify that TSS describes the Supported Elliptic Curves Extension and whether the required behaviour is performed by default or may be configured.

**FCS_TLSC_EXT.2.5**

232 The evaluator shall ensure that the TSS description required per FIA_X509_EXT.2.1 includes the use of client-side certificates for TLS mutual authentication.

#### 2.2.14.2 Guidance Documentation

**FCS_TLSC_EXT.2.1**

233 The evaluator shall also check the guidance documentation to ensure that it contains instructions on configuring the TOE so that TLS conforms to the description in the TSS.

**FCS_TLSC_EXT.2.2**

234 The evaluator shall verify that the AGD guidance includes instructions for setting the reference identifier to be used for the purposes of certificate validation in TLS.

**FCS_TLSC_EXT.2.4**

235 If the TSS indicates that the Supported Elliptic Curves Extension must be configured to meet the requirement, the evaluator shall verify that AGD guidance includes configuration of the Supported Elliptic Curves Extension.

**FCS_TLSC_EXT.2.5**

236 The evaluator shall verify that the AGD guidance required per FIA_X509_EXT.2.1 includes instructions for configuring the client-side certificates for TLS mutual authentication.

### 2.2.14.3 Tests

**FCS_TLSC_EXT.2.1**

237 Test 1: The evaluator shall establish a TLS connection using each of the ciphersuites specified by the requirement. This connection may be established as part of the establishment of a higher-level protocol, e.g., as part of an HTTPS session. It is sufficient to observe the successful negotiation of a ciphersuite to satisfy the intent of the test; it is not necessary to examine the characteristics of the encrypted traffic in an attempt to discern the ciphersuite being used (for example, that the cryptographic algorithm is 128-bit AES and not 256-bit AES).

238 Test 2: The evaluator shall attempt to establish the connection using a server with a server certificate that contains the Server Authentication purpose in the extendedKeyUsage field and verify that a connection is established. The evaluator will then verify that the client rejects an otherwise valid server certificate that lacks the Server Authentication purpose in the extendedKeyUsage field and a connection is not established. Ideally, the two certificates should be identical except for the extendedKeyUsage field.

239 Test 3: The evaluator shall send a server certificate in the TLS connection that the does not match the server-selected ciphersuite (for example, send a ECDSA certificate while using the TLS_RSA_WITH_AES_128_CBC_SHA ciphersuite.) The evaluator shall verify that the TOE disconnects after receiving the server's Certificate handshake message.

240 Test 4: The evaluator shall configure the server to select the TLS_NULL_WITH_NULL_NULL ciphersuite and verify that the client denies the connection. Test 2 in FCS_TLSS_EXT.1.1 or FCS_TLSS_EXT.2.1 can be used as a substitute for this test.

241 Test 5: The evaluator perform the following modifications to the traffic:

a) Change the TLS version selected by the server in the Server Hello to a non-supported TLS version (for example 1.3 represented by the two bytes 03 04) and verify that the client rejects the connection.

b) Modify at least one byte in the server's nonce in the Server Hello handshake message, and verify that the client rejects the Server Key Exchange handshake message (if using a DHE or ECDHE

ciphersuite) or that the server denies the client's Finished handshake message.

c) Modify the server's selected ciphersuite in the Server Hello handshake message to be a ciphersuite not presented in the Client Hello handshake message. The evaluator shall verify that the client rejects the connection after receiving the Server Hello.

d) Modify the signature block in the Server's Key Exchange handshake message, and verify that the client rejects the connection after receiving the Server Key Exchange message.

e) Modify a byte in the Server Finished handshake message, and verify that the client sends a fatal alert upon receipt and does not send any application data.

f) Send a garbled message from the Server after the Server has issued the ChangeCipherSpec message and verify that the client denies the connection.

**FCS_TLSC_EXT.2.2**

242 The evaluator shall configure the reference identifier according to the AGD guidance and perform the following tests during a TLS connection:

a) Test 1: The evaluator shall present a server certificate that does not contain an identifier in either the Subject Alternative Name (SAN) or Common Name (CN) that matches the reference identifier. The evaluator shall verify that the connection fails.

b) Test 2: The evaluator shall present a server certificate that contains a CN that matches the reference identifier, contains the SAN extension, but does not contain an identifier in the SAN that matches the reference identifier. The evaluator shall verify that the connection fails. The evaluator shall repeat this test for each supported SAN type.

c) Test 3: The evaluator shall present a server certificate that contains a CN that matches the reference identifier and does not contains the SAN extension. The evaluator shall verify that the connection succeeds.

d) Test 4: The evaluator shall present a server certificate that contains a CN that does not match the reference identifier but does contain an identifier in the SAN that matches. The evaluator shall verify that the connection succeeds.

e) Test 5: The evaluator shall perform the following wildcard tests with each supported type of reference identifier:

1) The evaluator shall present a server certificate containing a wildcard that is not in the left-most label of the presented

identifier (e.g. foo.*.example.com) and verify that the connection fails.

2) The evaluator shall present a server certificate containing a wildcard in the left-most label (e.g. *.example.com). The evaluator shall configure the reference identifier with a single left-most label (e.g. foo.example.com) and verify that the connection succeeds. The evaluator shall configure the reference identifier without a left-most label as in the certificate (e.g. example.com) and verify that the connection fails. The evaluator shall configure the reference identifier with two left-most labels (e.g. bar.foo.example.come) and verify that the connection fails.

f) Test 6: [conditional] If URI or Service name reference identifiers are supported, the evaluator shall configure the DNS name and the service identifier. The evaluator shall present a server certificate containing the correct DNS name and service identifier in the URIName or SRVName fields of the SAN and verify that the connection succeeds. The evaluator shall repeat this test with the wrong service identifier (but correct DNS name) and verify that the connection fails.

g) Test 7: [conditional] If pinned certificates are supported the evaluator shall present a certificate that does not match the pinned certificate and verify that the connection fails.

**FCS_TLSC_EXT.2.3**

243 Test 1: The evaluator shall demonstrate that using a certificate without a valid certification path results in the function failing. Using the administrative guidance, the evaluator shall then load a certificate or certificates needed to validate the certificate to be used in the function, and demonstrate that the function succeeds. . If the certificate is validated and a trusted channel is established, the test passes. The evaluator then shall delete one of the certificates, and show that the certificate is not validated and the trusted channel is not established.

**FCS_TLSC_EXT.2.4**

244 Test 1: The evaluator shall configure the server to perform an ECDHE key exchange in the TLS connection using a non-supported curve (for example P-192) and shall verify that the TOE disconnects after receiving the server's Key Exchange handshake message.

**FCS_TLSC_EXT.2.5**

245 Test 1: The evaluator shall perform the following modification to the traffic:

a) Configure the server to require mutual authentication and then modify a byte in a CA field in the Server's Certificate Request handshake message. The modified CA field must not be the CA used

to sign the client's certificate. The evaluator shall verify the connection fails.

### 2.2.15 FCS_TLSS_EXT.1 Extended: TLS Server Protocol

#### 2.2.15.1 TSS

**FCS_TLSS_EXT.1.1**

246 The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that the ciphersuites supported are specified. The evaluator shall check the TSS to ensure that the ciphersuites specified are identical to those listed for this component.

**FCS_TLSS_EXT.1.2**

247 The evaluator shall verify that the TSS contains a description of the denial of old SSL and TLS versions.

**FCS_TLSS_EXT.1.3**

248 The evaluator shall verify that the TSS describes the key agreement parameters of the server key exchange message.

#### 2.2.15.2 Guidance Documentation

**FCS_TLSS_EXT.1.1**

249 The evaluator shall also check the guidance documentation to ensure that it contains instructions on configuring the TOE so that TLS conforms to the description in the TSS (for instance, the set of ciphersuites advertised by the TOE may have to be restricted to meet the requirements).

**FCS_TLSS_EXT.1.2**

250 The evaluator shall verify that any configuration necessary to meet the requirement must be contained in the AGD guidance.

**FCS_TLSS_EXT.1.3**

251 The evaluator shall verify that any configuration necessary to meet the requirement must be contained in the AGD guidance.

#### 2.2.15.3 Tests

**FCS_TLSS_EXT.1.1**

252 Test 1: The evaluator shall establish a TLS connection using each of the ciphersuites specified by the requirement. This connection may be established as part of the establishment of a higher-level protocol, e.g., as part of an HTTPS session. It is sufficient to observe the successful negotiation of a ciphersuite to satisfy the intent of the test; it is not necessary to examine the characteristics of the encrypted traffic in an attempt to discern

the ciphersuite being used (for example, that the cryptographic algorithm is 128-bit AES and not 256-bit AES).

253 Test 2: The evaluator shall send a Client Hello to the server with a list of ciphersuites that does not contain any of the ciphersuites in the server's ST and verify that the server denies the connection. Additionally, the evaluator shall send a Client Hello to the server containing only the TLS_NULL_WITH_NULL_NULL ciphersuite and verify that the server denies the connection.

254 Test 3: The evaluator shall use a client to send a key exchange message in the TLS connection that the does not match the server-selected ciphersuite (for example, send an ECDHE key exchange while using the TLS_RSA_WITH_AES_128_CBC_SHA ciphersuite or send a RSA key exchange while using one of the ECDSA ciphersuites.) The evaluator shall verify that the TOE disconnects after the receiving the key exchange message.

255 Test 4: The evaluator shall perform the following modifications to the traffic:

a) Modify at a byte in the client's nonce in the Client Hello handshake message, and verify that the server rejects the client's Certificate Verify handshake message (if using mutual authentication) or that the server denies the client's Finished handshake message.

b) Modify the signature block in the Client's Key Exchange handshake message, and verify that the server rejects the client's Certificate Verify handshake message (if using mutual authentication) or that the server denies the client's Finished handshake message.

c) Modify a byte in the Client Finished handshake message, and verify that the server rejects the connection and does not send any application data.

d) After generating a fatal alert by sending a Finished message from the client before the client sends a ChangeCipherSpec message, send a Client Hello with the session identifier from the previous test, and verify that the server denies the connection.

e) Send a garbled message from the client after the client has issued the ChangeCipherSpec message and verify that the Server denies the connection.

**FCS_TLSS_EXT.1.2**

256 The evaluator shall send a Client Hello requesting a connection with version SSL 1.0 and verify that the server denies the connection. The evaluator shall repeat this test with SSL 2.0, SSL 3.0, TLS 1.0, and any selected TLS versions.

**FCS_TLSS_EXT.1.3**

257 The evaluator shall attempt a connection using an ECDHE ciphersuite and a configured curve and, using a packet analyzer, verify that the key agreement parameters in the Key Exchange message are the ones configured. (Determining that the size matches the expected size for the configured curve is sufficient.) The evaluator shall repeat this test for each supported NIST Elliptic Curve and each supported Diffie-Hellman key size.

### 2.2.16 FCS_TLSS_EXT.2 Extended: TLS Server Protocol with mutual authentication

#### 2.2.16.1 TSS

**FCS_TLSS_EXT.2.1**

258 The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that the ciphersuites supported are specified. The evaluator shall check the TSS to ensure that the ciphersuites specified are identical to those listed for this component.

**FCS_TLSS_EXT.2.2**

259 The evaluator shall verify that the TSS contains a description of the denial of old SSL and TLS versions.

**FCS_TLSS_EXT.2.3**

260 The evaluator shall verify that the TSS describes the key agreement parameters of the server key exchange message.

**FCS_TLSS_EXT.2.4 and FCS_TLSS_EXT.2.5**

261 The evaluator shall ensure that the TSS description required per FIA_X509_EXT.2.1 includes the use of client-side certificates for TLS mutual authentication.

**FCS_TLSS_EXT.2.6**

262 The evaluator shall verify that the TSS describes how the DN or SAN in the certificate is compared to the expected identifier.

#### 2.2.16.2 Guidance Documentation

**FCS_TLSS_EXT.2.1**

263 The evaluator shall also check the guidance documentation to ensure that it contains instructions on configuring the TOE so that TLS conforms to the description in the TSS (for instance, the set of ciphersuites advertised by the TOE may have to be restricted to meet the requirements).

**FCS_TLSS_EXT.2.2**

264 The evaluator shall verify that any configuration necessary to meet the requirement must be contained in the AGD guidance.

**FCS_TLSS_EXT.2.3**

265 The evaluator shall verify that any configuration necessary to meet the requirement must be contained in the AGD guidance.

**FCS_TLSS_EXT.2.4 and FCS_TLSS_EXT.2.5**

266 The evaluator shall verify that the AGD guidance required per FIA_X509_EXT.2.1 includes instructions for configuring the client-side certificates for TLS mutual authentication.

**FCS_TLSS_EXT.2.6**

267 If the DN is not compared automatically to the Domain Name or IP address, username, or email address, then the evaluator shall ensure that the AGD guidance includes configuration of the expected DN or the directory server for the connection.

#### 2.2.16.3 Tests

**FCS_TLSS_EXT.2.1**

268 Test 1: The evaluator shall establish a TLS connection using each of the ciphersuites specified by the requirement. This connection may be established as part of the establishment of a higher-level protocol, e.g., as part of an HTTPS session. It is sufficient to observe the successful negotiation of a ciphersuite to satisfy the intent of the test; it is not necessary to examine the characteristics of the encrypted traffic in an attempt to discern the ciphersuite being used (for example, that the cryptographic algorithm is 128-bit AES and not 256-bit AES).

269 Test 2: The evaluator shall send a Client Hello to the server with a list of ciphersuites that does not contain any of the ciphersuites in the server's ST and verify that the server denies the connection. Additionally, the evaluator shall send a Client Hello to the server containing only the TLS_NULL_WITH_NULL_NULL ciphersuite and verify that the server denies the connection.

270 Test 3: The evaluator shall use a client to send a key exchange message in the TLS connection that the does not match the server-selected ciphersuite (for example, send an ECDHE key exchange while using the TLS_RSA_WITH_AES_128_CBC_SHA ciphersuite or send a RSA key exchange while using one of the ECDSA ciphersuites.) The evaluator shall verify that the TOE disconnects after the receiving the key exchange message.

271 Test 4: The evaluator shall perform the following modifications to the traffic:

a) Modify at a byte in the client's nonce in the Client Hello handshake message, and verify that the server rejects the client's Certificate Verify handshake message (if using mutual authentication) or that the server denies the client's Finished handshake message.

b) Modify the signature block in the Client's Key Exchange handshake message, and verify that the server rejects the client's Certificate Verify handshake message (if using mutual authentication) or that the server denies the client's Finished handshake message.

c) Modify a byte in the Client Finished handshake message, and verify that the server rejects the connection and does not send any application data.

d) After generating a fatal alert by sending a Finished message from the client before the client sends a ChangeCipherSpec message, send a Client Hello with the session identifier from the previous test, and verify that the server denies the connection.

e) Send a garbled message from the client after the client has issued the ChangeCipherSpec message and verify that the Server denies the connection.

**FCS_TLSS_EXT.2.2**

272 The evaluator shall send a Client Hello requesting a connection with version SSL 1.0 and verify that the server denies the connection. The evaluator shall repeat this test with SSL 2.0, SSL 3.0, TLS 1.0, and any selected TLS versions.

**FCS_TLSS_EXT.2.3**

273 The evaluator shall attempt a connection using an ECDHE ciphersuite and a configured curve and, using a packet analyzer, verify that the key agreement parameters in the Key Exchange message are the ones configured. (Determining that the size matches the expected size for the configured curve is sufficient.) The evaluator shall repeat this test for each supported NIST Elliptic Curve and each supported Diffie-Hellman key size.

**FCS_TLSS_EXT.2.4 and FCS_TLSS_EXT.2.5**

274 Test 1: The evaluator shall configure the server to send a certificate request to the client and shall attempt a connection without sending a certificate from the client. The evaluator shall verify that the connection is denied.

275 Test 2: The evaluator shall configure the server to send a certificate request to the client without the supported_signature_algorithm used by the client's certificate. The evaluator shall attempt a connection using the client certificate and verify that the connection is denied.

276 Test 3: The evaluator shall demonstrate that using a certificate without a valid certification path results in the function failing. Using the administrative guidance, the evaluator shall then load a certificate or certificates needed to validate the certificate to be used in the function, and demonstrate that the function succeeds. The evaluator then shall delete one of the certificates, and show that the function fails.

277 Test 4: The evaluator shall configure the client to send a certificate that does not chain to one of the Certificate Authorities (either a Root or Intermediate CA) in the server's Certificate Request message. The evaluator shall verify that the attempted connection is denied.

278 Test 5: The evaluator shall configure the client to send a certificate with the Client Authentication purpose in the extendedKeyUsage field and verify that the server accepts the attempted connection. The evaluator shall repeat this test without the Client Authentication purpose and shall verify that the server denies the connection. Ideally, the two certificates should be identical except for the Client Authentication purpose.

279 Test 6: The evaluator shall perform the following modifications to the traffic:

a) Configure the server to require mutual authentication and then modify a byte in the client's certificate. The evaluator shall verify that the server rejects the connection.

b) Configure the server to require mutual authentication and then modify a byte in the client's Certificate Verify handshake message. The evaluator shall verify that the server rejects the connection.

**FCS_TLSS_EXT.2.6**

280 The evaluator shall send a client certificate with an identifier that does not match an expected identifier and verify that the server denies the connection.

## 2.3 Identification and Authentication (FIA)

### 2.3.1 FIA_PMG_EXT.1 Password Management

#### 2.3.1.1 Guidance Documentation

281 The evaluator shall examine the guidance documentation to determine that it provides guidance to security administrators on the composition of strong passwords, and that it provides instructions on setting the minimum password length.

#### 2.3.1.2 Tests

282 The evaluator shall perform the following tests.

a) Test 1: The evaluator shall compose passwords that either meet the requirements, or fail to meet the requirements, in some way. For each password, the evaluator shall verify that the TOE supports the password. While the evaluator is not required (nor is it feasible) to test all possible compositions of passwords, the evaluator shall ensure that all characters, rule characteristics, and a minimum length listed in the requirement are supported, and justify the subset of those characters chosen for testing.

### 2.3.2 FIA_UIA_EXT.1 User Identification and Authentication

#### 2.3.2.1 TSS

283 The evaluator shall examine the TSS to determine that it describes the logon process for each logon method (local, remote (HTTPS, SSH, etc.)) supported for the product. This description shall contain information pertaining to the credentials allowed/used, any protocol transactions that take place, and what constitutes a "successful logon".

#### 2.3.2.2 Guidance Documentation

284 The evaluator shall examine the guidance documentation to determine that any necessary preparatory steps (e.g., establishing credential material such as pre- shared keys, tunnels, certificates, etc.) to logging in are described. For each supported the login method, the evaluator shall ensure the guidance documentation provides clear instructions for successfully logging on. If configuration is necessary to ensure the services provided before login are limited, the evaluator shall determine that the guidance documentation provides sufficient instruction on limiting the allowed services.

#### 2.3.2.3 Tests

285 The evaluator shall perform the following tests for each method by which administrators access the TOE (local and remote), as well as for each type of credential supported by the login method:

a) Test 1: The evaluator shall use the guidance documentation to configure the appropriate credential supported for the login method. For that credential/login method, the evaluator shall show that providing correct I&A information results in the ability to access the system, while providing incorrect information results in denial of access.

b) Test 2: The evaluator shall configure the services allowed (if any) according to the guidance documentation, and then determine the services available to an external remote entity. The evaluator shall determine that the list of services available is limited to those specified in the requirement.

c) Test 3: For local access, the evaluator shall determine what services are available to a local administrator prior to logging in, and make sure this list is consistent with the requirement.

### 2.3.3 FIA_UAU_EXT.2 Password-based Authentication Mechanism

286 Evaluation Activities for this requirement are covered under those for FIA_UIA_EXT.1. If other authentication mechanisms are specified, the evaluator shall include those methods in the activities for FIA_UIA_EXT.1.

### 2.3.4 FIA_UAU.7 Protected Authentication Feedback

#### 2.3.4.1 Tests

287 The evaluator shall perform the following test for each method of local login allowed:

a) Test 1: The evaluator shall locally authenticate to the TOE. While making this attempt, the evaluator shall verify that at most obscured feedback is provided while entering the authentication information.

### 2.3.5 FIA_X509_EXT.1 X.509 Certificate Validation

#### 2.3.5.1 TSS

288 The evaluator shall ensure the TSS describes where the check of validity of the certificates takes place. The evaluator ensures the TSS also provides a description of the certificate path validation algorithm.

#### 2.3.5.2 Tests

289 The evaluator shall perform the following tests for FIA_X509_EXT.1.1:

a) Test 1: The evaluator shall demonstrate that validating a certificate without a valid certification path results in the function failing. The evaluator shall then load a certificate or certificates as trusted CAs needed to validate the certificate to be used in the function, and demonstrate that the function succeeds. The evaluator shall then delete one of the certificates, and show that the function fails.

b) Test 2: The evaluator shall demonstrate that validating an expired certificate results in the function failing.

c) Test 3: The evaluator shall test that the TOE can properly handle revoked certificates--conditional on whether CRL or OCSP is selected; if both are selected, then a test shall be performed for each method. The evaluator shall test revocation of the TOE certificate and revocation of the TOE intermediate CA certificate i.e. the intermediate CA certificate should be revoked by the root CA. The evaluator shall ensure that a valid certificate is used, and that the validation function succeeds. The evaluator then attempts the test with a certificate that has been revoked (for each method chosen in the selection) to ensure when the certificate is no longer valid that the validation function fails.

d) Test 4: If OCSP is selected, the evaluator shall configure the OCSP server or use a man-in-the-middle tool to present a certificate that does not have the OCSP signing purpose and verify that validation of the OCSP response fails. If CRL is selected, the evaluator shall configure the CA to sign a CRL with a certificate that does not have

the cRLsign key usage bit set, and verify that validation of the CRL fails.

e) Test 5: The evaluator shall modify any byte in the first eight bytes of the certificate and demonstrate that the certificate fails to validate. (The certificate will fail to parse correctly.)

f) Test 6: The evaluator shall modify any byte in the last byte of the certificate and demonstrate that the certificate fails to validate. (The signature on the certificate will not validate.)

g) Test 7: The evaluator shall modify any byte in the public key of the certificate and demonstrate that the certificate fails to validate. (The hash of the certificate will not validate.)

290 The evaluator shall perform the following tests for FIA_X509_EXT.1.2. The tests described must be performed in conjunction with the other certificate services assurance activities, including the functions in FIA_X509_EXT.2.1. The tests for the extendedKeyUsage rules are performed in conjunction with the uses that require those rules.

291 The evaluator shall create a chain of at least four certificates: the node certificate to be tested, two intermediate CAs, and the self-signed Root CA.

a) Test 1: The evaluator shall construct a certificate path, such that the certificate of the CA issuing the TOE's certificate does not contain the basicConstraints extension. The validation of the certificate path fails.

b) Test 2: The evaluator shall construct a certificate path, such that the certificate of the CA issuing the TOE's certificate has the cA flag in the basicConstraints extension set to FALSE. The validation of the certificate path fails.

c) Test 3: The evaluator shall construct a certificate path, such that the certificate of the CA issuing the TOE's certificate has the cA flag in the basicConstraints extension set to TRUE. The validation of the certificate path succeeds.

### 2.3.6 FIA_X509_EXT.2 X.509 Certificate Authentication

#### 2.3.6.1 TSS

292 The evaluator shall check the TSS to ensure that it describes how the TOE chooses which certificates to use, and any necessary instructions in the administrative guidance for configuring the operating environment so that the TOE can use the certificates.

293 The evaluator shall examine the TSS to confirm that it describes the behaviour of the TOE when a connection cannot be established during the

validity check of a certificate used in establishing a trusted channel. The evaluator shall verify that any distinctions between trusted channels are described. If the requirement that the administrator is able to specify the default action, then the evaluator shall ensure that the guidance documentation contains instructions on how this configuration action is performed.

#### 2.3.6.2 Tests

294 The evaluator shall perform the following test for each trusted channel:

295 The evaluator shall demonstrate that using a valid certificate that requires certificate validation checking to be performed in at least some part by communicating with a non-TOE IT entity. The evaluator shall then manipulate the environment so that the TOE is unable to verify the validity of the certificate, and observe that the action selected in FIA_X509_EXT.2.2 is performed. If the selected action is administrator-configurable, then the evaluator shall follow the guidance documentation to determine that all supported administrator-configurable options behave in their documented manner.

### 2.3.7 FIA_X509_EXT.3 Extended: X509 Certificate Requests

#### 2.3.7.1 TSS

296 If the ST author selects "device-specific information", the evaluator shall verify that the TSS contains a description of the device-specific fields used in certificate requests.

#### 2.3.7.2 Guidance Documentation

**297** The evaluator shall check to ensure that the guidance documentation contains instructions on requesting certificates from a CA, including generation of a Certificate Request Message. If the ST author selects "Common Name", "Organization", "Organizational Unit", or "Country", the evaluator shall ensure that this guidance includes instructions for establishing these fields before creating the certificate request message.

#### 2.3.7.3 Tests

298 The evaluator shall perform the following tests:

a) Test 1: The evaluator shall use the guidance documentation to cause the TOE to generate a certificate request message. The evaluator shall capture the generated message and ensure that it conforms to the format specified. The evaluator shall confirm that the certificate request provides the public key and other required information, including any necessary user-input information.

b) Test 2: The evaluator shall demonstrate that validating a certificate response message without a valid certification path results in the function failing. The evaluator shall then load a certificate or certificates as trusted CAs needed to validate the certificate response message, and demonstrate that the function succeeds. The evaluator shall then delete one of the certificates, and show that the function fails.

## 2.4 Security management (FMT)

### 2.4.1 FMT_MOF.1(1)/TrustedUpdate

#### 2.4.1.1 Tests

299 The evaluator shall try to perform the update using a legitimate update image without prior authentication as security administrator (either by authentication as a user with no administrator privileges or without user authentication at all – depending on the configuration of the TOE). This test should fail.

300 The evaluator shall try to perform the update with prior authentication as security administrator using a legitimate update image. This test should pass. This test case should be covered by the tests for FPT_TUD_EXT.1 already.

### 2.4.2 FMT_MOF.1(2)/TrustedUpdate

#### 2.4.2.1 Tests

301 The evaluator shall try to enable and disable automatic checking for updates or automatic updates (whichever is supported by the TOE) without prior authentication as security administrator (either by authentication as a user with no administrator privileges or without user authentication at all – depending on the configuration of the TOE). This test should fail.

302 The evaluator shall try to enable and disable automatic checking for updates or automatic updates (whichever is supported by the TOE) with prior authentication as security administrator. This test should pass.

### 2.4.3 FMT_MOF.1(1)/Audit

#### 2.4.3.1 Tests

303 The evaluator shall try to modify all parameters for configuration of the transmission protocol for transmission of audit data to an external IT entity without prior authentication as security administrator (either by authentication as a user with no administrator privileges or without user authentication at all – depending on the configuration of the TOE). This test should fail.

304 The evaluator shall try to modify all parameters for configuration of the transmission protocol for transmission of audit data to an external IT entity with prior authentication as security administrator. The effects of the modifications should be confirmed.

305 The evaluator does not necessarily have to test all possible values of all parameters for configuration of the transmission protocol for transmission of audit data to an external IT entity but at least one allowed value per configurable parameter.

### 2.4.4 FMT_MOF.1(2)/Audit

#### 2.4.4.1 Tests

306 The evaluator shall try to modify all parameters for configuration of the handling of audit data without prior authentication as security administrator (either by authentication as a user with no administrator privileges or without user authentication at all – depending on the configuration of the TOE). This test should fail. The term 'handling of audit data' refers to the different options for selection and assignments in SFRs FAU_STG_EXT.1.2, FAU_STG_EXT.1.3 and FAU_STG_EXT.2.

307 The evaluator shall try to modify all parameters for configuration of the handling of audit data with prior authentication as security administrator. The effects of the modifications should be confirmed. The term 'handling of audit data' refers to the different options for selection and assignments in SFRs FAU_STG_EXT.1.2, FAU_STG_EXT.1.3 and FAU_STG_EXT.2.

308 The evaluator does not necessarily have to test all possible values of all parameters for configuration of the handling of audit data but at least one allowed value per configurable parameter.

### 2.4.5 FMT_MOF.1(1)/AdminAct

#### 2.4.5.1 Tests

309 The evaluator shall try to perform at least one of the related actions without prior authentication as security administrator (either by authentication as a user with no administrator privileges or without user authentication at all – depending on the configuration of the TOE). These attempts should fail.

310 The evaluator shall try to perform at least one of the related actions with prior authentication as security administrator. These attempts should succeed.

### 2.4.6 FMT_MOF.1(2)/AdminAct

#### 2.4.6.1 Tests

311 The evaluator shall try to perform at least one of the related actions without prior authentication as security administrator (either by authentication as a

user with no administrator privileges or without user authentication at all – depending on the configuration of the TOE). These attempts should fail.

312 The evaluator shall try to perform at least one of the related actions with prior authentication as security administrator. These attempts should succeed.

### 2.4.7 FMT_MOF.1/LocSpace Management of security functions behaviour

#### 2.4.7.1 Tests

313 The evaluator shall try to perform at least one of the related actions without prior authentication as security administrator (either by authentication as a user with no administrator privileges or without user authentication at all – depending on the configuration of the TOE). These attempts should fail.

314 The evaluator shall try to perform at least one of the related actions with prior authentication as security administrator. These attempts should succeed.

### 2.4.8 FMT_MTD.1 Management of TSF Data

#### 2.4.8.1 TSS

315 The evaluator shall examine the TSS to determine that, for each administrative function identified in the guidance documentation; those that are accessible through an interface prior to administrator log-in are identified. For each of these functions, the evaluator shall also confirm that the TSS details how the ability to manipulate the TSF data through these interfaces is disallowed for non-administrative users.

#### 2.4.8.2 Guidance Documentation

316 The evaluator shall review the guidance documentation to determine that each of the TSF-data-manipulating functions implemented in response to the requirements of the cPP is identified, and that configuration information is provided to ensure that only administrators have access to the functions.

### 2.4.9 FMT_MTD.1/AdminAct Management of TSF Data

#### 2.4.9.1 Tests

317 The evaluator shall try to perform at least one of the related actions without prior authentication as security administrator (either by authentication as a user with no administrator privileges or without user authentication at all – depending on the configuration of the TOE). This test should fail.

318 The evaluator shall try to perform at least one of the related actions with prior authentication as security administrator. This test should pass.

### 2.4.10 FMT_SMF.1 Specification of Management Functions

319 The security management functions for FMT_SMF.1 are distributed throughout the cPP and are included as part of the requirements in FTA_TAB.1, FTA_SSL.3, FTA_SSL.4, FMT_MOF.1(1)/TrustedUpdate, FMT_MOF.1(2)/TrustedUpdate (if included in the ST), FIA_X509_EXT.2.2 & FPT_TUD_EXT.2.2 (if included in the ST and if they include an administrator-configurable action), FMT_MOF.1(1)/Audit, FMT_MOF.1(2)/Audit, FMT_MOF.1.1(1)/AdminAct, FMT_MOF.1.1(2)/AdminAct and FMT_MOF.1/LocSpace (for all of these SFRs that are included in the ST), FMT_MTD, FPT_TST_EXT, and any cryptographic management functions specified in the reference standards. Compliance to these requirements satisfies compliance with FMT_SMF.1.

### 2.4.11 FMT_SMR.2 Restrictions on security roles

#### 2.4.11.1 Guidance Documentation

320 The evaluator shall review the guidance documentation to ensure that it contains instructions for administering the TOE both locally and remotely, including any configuration that needs to be performed on the client for remote administration.

#### 2.4.11.2 Tests

321 In the course of performing the testing activities for the evaluation, the evaluator shall use all supported interfaces, although it is not necessary to repeat each test involving an administrative action with each interface. The evaluator shall ensure, however, that each supported method of administering the TOE that conforms to the requirements of this cPP be tested; for instance, if the TOE can be administered through a local hardware interface; SSH; and TLS/HTTPS; then all three methods of administration must be exercised during the evaluation team's test activities.

## 2.5 Protection of the TSF (FPT)

### 2.5.1 FPT_SKP_EXT.1 Protection of TSF Data (for reading of all symmetric keys)

#### 2.5.1.1 TSS

322 The evaluator shall examine the TSS to determine that it details how any pre-shared keys, symmetric keys, and private keys are stored and that they are unable to be viewed through an interface designed specifically for that purpose, as outlined in the application note. If these values are not stored in plaintext, the TSS shall describe how they are protected/obscured.

### 2.5.2 FPT_APW_EXT.1 Protection of Administrator Passwords

#### 2.5.2.1 TSS

323 The evaluator shall examine the TSS to determine that it details all authentication data that are subject to this requirement, and the method used to obscure the plaintext password data when stored. The TSS shall also detail passwords are stored in such a way that they are unable to be viewed through an interface designed specifically for that purpose, as outlined in the application note.

### 2.5.3 FPT_TST_EXT.1 TSF testing

#### 2.5.3.1 TSS

324 The evaluator shall examine the TSS to ensure that it details the self tests that are run by the TSF; this description should include an outline of what the tests are actually doing (e.g., rather than saying "memory is tested", a description similar to "memory is tested by writing a value to each memory location and reading it back to ensure it is identical to what was written" shall be used). The evaluator shall ensure that the TSS makes an argument that the tests are sufficient to demonstrate that the TSF is operating correctly.

#### 2.5.3.2 Guidance Documentation

325 The evaluator shall also ensure that the guidance documentation describes the possible errors that may result from such tests, and actions the administrator should take in response; these possible errors shall correspond to those described in the TSS.

#### 2.5.3.3 Tests

326 Future versions of this cPP will mandate a clearly defined minimum set of self tests. But also for this version of the cPP it is expected that at least the following tests are performed:

a) Verification of the integrity of the firmware and executable software of the TOE

b) Verification of the correct operation of the cryptographic functions necessary to fulfill any of the SFRs.

327 Although formal compliance is not mandated, the self tests performed should aim for a level of confidence comparable to:

a) FIPS 140-2, chap. 4.9.1, Software/firmware integrity test for the verification of the integrity of the firmware and executable software.

b) FIPS 140-2, chap. 4.9.1, Cryptographic algorithm test for the verification of the correct operation of cryptographic functions.

Alternatively, national requirements of any CCRA member state for the security evaluation of cryptographic functions should be considered as appropriate.

328 The evaluator shall verify that the self tests described above are either carried out during initial start-up and that the developer has justified any deviation from this (if applicable).

### 2.5.4 FPT_TST_EXT.2 Self tests based on certificates

#### 2.5.4.1 Tests

329 The evaluator shall verify that the self test mechanism includes a certificate validation according to FIA_X509_EXT.1 and a check for the Code Signing purpose in the extendedKeyUsage.

330 The evaluator shall use an invalid certificate and perform the self test. This test should fail. The evaluator shall use a certificate that does not have the Code Signing purpose and verify that the self test fails. The evaluator shall repeat the test using a valid certificate and a certificate that contains the Code Signing purpose and verify that the self test succeeds. Testing for this element is performed in conjunction with the assurance activities for FPT_TST_EXT.1.

### 2.5.5 FPT_TUD_EXT.1 Trusted Update

#### 2.5.5.1 TSS

331 The evaluator shall verify that the TSS describes all TSF software update mechanisms for updating the system software. The evaluator shall verify that the description includes a digital signature verification of the software before installation and that installation fails if the verification fails. Alternatively an approach using a published hash can be used. In this case the TSS shall detail this mechanism instead of the digital signature verification mechanism. The evaluator shall verify that the TSS describes the method by which the digital signature or published hash is verified to include how the candidate updates are obtained, the processing associated with verifying the digital signature or published hash of the update, and the actions that take place for both successful and unsuccessful signature verification or published hash verification.

332 If the ST author indicates that a certificate-based mechanism is used for software update digital signature verification, the evaluator shall verify that the TSS contains a description of how the certificates are contained on the device. The evaluator also ensures that the TSS (or guidance documentation) describes how the certificates are installed/updated/selected, if necessary.

#### 2.5.5.2 Guidance Documentation

333 The evaluator shall verify that the guidance documentation describes how the verification of the authenticity of the update is performed (digital signature verification or verification of published hash). The description shall include the procedures for successful and unsuccessful verification. The description shall correspond to the description in the TSS.

#### 2.5.5.3 Tests

334 The evaluator shall perform the following tests:

a) Test 1: The evaluator performs the version verification activity to determine the current version of the product as well as the most recently installed version (should be the same version before updating). The evaluator obtains a legitimate update using procedures described in the guidance documentation and verifies that it is successfully installed on the TOE. For some TOEs loading the update onto the TOE and activation of the update are separate steps ('activation' could be performed e.g. by a distinct activation step or by rebooting the device). In that case the evaluator verifies after loading the update onto the TOE but before activation of the update that the current version of the product did not change but the most recently installed version has changed to the new product version. After the update, the evaluator performs the version verification activity again to verify the version correctly corresponds to that of the update and that current version of the product and most recently installed version match again.

b) Test 2: The evaluator performs the version verification activity to determine the current version of the product as well as the most recently installed version (should be the same version before updating). The evaluator obtains or produces illegitimate updates as defined below, and attempts to install them on the TOE. The evaluator verifies that the TOE rejects all of the illegitimate updates. The evaluator performs this test using all of the following forms of illegitimate updates:

1) A modified version (e.g. using a hex editor) of a legitimately signed update (if digital signatures are used) or a version that does not match the published hash (if published hashes are used)

2) An image that has not been signed (if digital signatures are used) or an image without published hash (if published hashes are used)

3) An image signed with an invalid signature (e.g. by using a different key as expected for creating the signature or by manual modification of a legitimate signature) (only if digital signatures are used).

4) The handling of version information of the most recently installed version might differ between different TOEs. Depending on the point in time when the attempted update is rejected, the most recently installed version might or might not be updated. The evaluator shall verify that the TOE handles the most recently installed version information for that case as described in the guidance documentation. After the TOE has rejected the update the evaluator shall verify, that both, current version and most recently installed version, reflect the same version information as prior to the update attempt.

335 The evaluator shall perform the Tests 1 and 2 for all methods supported (manual updates, automatic checking for updates, automatic updates).

### 2.5.6 FPT_TUD_EXT.2 Trusted Update based on certificates

#### 2.5.6.1 TSS

336 The evaluator shall verify that the TSS describes how the TOE reacts if X.509 certificates are used for trusted updates and the administrator attempts to perform the trusted update using an expired certificate.

#### 2.5.6.2 Guidance Documentation

337 The evaluator shall verify that the guidance documentation describes how the TOE reacts if X.509 certificates are used for trusted updates and the administrator attempts to perform the trusted update using an expired certificate. The description shall correspond to the description in the TSS.

#### 2.5.6.3 Tests

338 The evaluator shall verify that the update mechanism includes a certificate validation according to FIA_X509_EXT.1 and a check for the Code Signing purpose in the extendedKeyUsage.

339 The evaluator shall digitally sign the update with an invalid certificate and verify that update installation fails. The evaluator shall digitally sign the application with a certificate that does not have the Code Signing purpose and verify that application installation fails. The evaluator shall repeat the test using a valid certificate and a certificate that contains the Code Signing purpose and verify that the application installation succeeds. The evaluator shall use a previously valid but expired certificate and verifies that the TOE reacts as described in the TSS and the guidance documentation. Testing for this element is performed in conjunction with the assurance activities for FPT_TUD_EXT.1.

### 2.5.7 FPT_STM.1 Reliable Time Stamps

#### 2.5.7.1 TSS

340 The evaluator shall examine the TSS to ensure that it lists each security function that makes use of time. The TSS provides a description of how the time is maintained and considered reliable in the context of each of the time related functions.

341 The evaluator examines the guidance documentation to ensure it instructs the administrator how to set the time. If the TOE supports the use of an NTP server, the guidance documentation instructs how a communication path is established between the TOE and the NTP server, and any configuration of the NTP client on the TOE to support this communication.

#### 2.5.7.2 Tests

342 The evaluator shall perform the following tests:

a) Test 1: The evaluator uses the guidance documentation to set the time. The evaluator shall then use an available interface to observe that the time was set correctly.

b) Test 2: If the TOE supports the use of an NTP server; the evaluator shall use the guidance documentation to configure the NTP client on the TOE, and set up a communication path with the NTP server. The evaluator will observe that the NTP server has set the time to what is expected. If the TOE supports multiple protocols for establishing a connection with the NTP server, the evaluator shall perform this test using each supported protocol claimed in the guidance documentation.

343 If the audit component of the TOE consists of several parts with independent time information, then the evaluator shall verify that the time information between the different parts are either synchronized or that it is possible for all audit information to relate the time information of the different part to one base information unambiguously.

### 2.5.8 FPT_FLS.1/LocSpace Failure with preservation of secure state

#### 2.5.8.1 Tests

344 The evaluator shall perform a test that the local storage space for audit data is not full (e.g. by executing an action that is logged and verifying that the audit data is updated accordingly). The evaluator shall test that the security functions are running properly (some sampling may be required here). Then the auditor shall execute activities that are logged until the local storage space for audit data is full. The evaluator shall verify that the security functions are no longer working or are no longer accessible. The security functions necessary to preserve the secure state according to

FPT_FLS.1/Local Audit Storage Space Full shall be regarded as an exception to this rule, since they have to work properly to fulfil the requirement itself. If the evaluator has used sampling for the verification that the security functions did run properly when the local space for audit data was not full, then the evaluator shall verify for the same security functions that they have stopped working after the local storage space for audit data is full.

## 2.6 TOE Access (FTA)

### 2.6.1 FTA_SSL_EXT.1 TSF-initiated Session Locking

#### 2.6.1.1 Tests

345 The evaluator shall perform the following test:

a) Test 1: The evaluator follows the guidance documentation to configure several different values for the inactivity time period referenced in the component. For each period configured, the evaluator establishes a local interactive session with the TOE. The evaluator then observes that the session is either locked or terminated after the configured time period. If locking was selected from the component, the evaluator then ensures that re-authentication is needed when trying to unlock the session.

### 2.6.2 FTA_SSL.3 TSF-initiated Termination

#### 2.6.2.1 Tests

346 The evaluator shall perform the following test:

a) Test 1: The evaluator follows the guidance documentation to configure several different values for the inactivity time period referenced in the component. For each period configured, the evaluator establishes a remote interactive session with the TOE. The evaluator then observes that the session is terminated after the configured time period.

### 2.6.3 FTA_SSL.4 User-initiated Termination

#### 2.6.3.1 Tests

347 The evaluator shall perform the following tests:

a) Test 1: The evaluator initiates an interactive local session with the TOE. The evaluator then follows the guidance documentation to exit or log off the session and observes that the session has been terminated.

b) Test 2: The evaluator initiates an interactive remote session with the TOE. The evaluator then follows the guidance documentation to

exit or log off the session and observes that the session has been terminated.

### 2.6.4 FTA_TAB.1 Default TOE Access Banners

#### 2.6.4.1 TSS

348 The evaluator shall check the TSS to ensure that it details each method of access (local and remote) available to the administrator (e.g., serial port, SSH, HTTPS).

#### 2.6.4.2 Tests

349 The evaluator shall also perform the following test:

a) Test 1: The evaluator follows the guidance documentation to configure a notice and consent warning message. The evaluator shall then, for each method of access specified in the TSS, establish a session with the TOE. The evaluator shall verify that the notice and consent warning message is displayed in each instance.

## 2.7 Trusted path/channels (FTP)

### 2.7.1 FTP_ITC.1 Inter-TSF trusted channel

#### 2.7.1.1 TSS

350 The evaluator shall examine the TSS to determine that, for all communications with authorized IT entities identified in the requirement, each communications mechanism is identified in terms of the allowed protocols for that IT entity. The evaluator shall also confirm that all protocols listed in the TSS are specified and included in the requirements in the ST.

#### 2.7.1.2 Guidance Documentation

351 The evaluator shall confirm that the guidance documentation contains instructions for establishing the allowed protocols with each authorized IT entity, and that it contains recovery instructions should a connection be unintentionally broken.

#### 2.7.1.3 Tests

352 The evaluator shall perform the following tests:

a) Test 1: The evaluators shall ensure that communications using each protocol with each authorized IT entity is tested during the course of the evaluation, setting up the connections as described in the guidance documentation and ensuring that communication is successful.

b) Test 2: For each protocol that the TOE can initiate as defined in the requirement, the evaluator shall follow the guidance documentation

to ensure that in fact the communication channel can be initiated from the TOE.

c) Test 3: The evaluator shall ensure, for each communication channel with an authorized IT entity, the channel data is not sent in plaintext.

d) Test 4: The evaluators shall, for each protocol associated with each authorized IT entity tested during test 1, the connection is physically interrupted. The evaluator shall ensure that when physical connectivity is restored, communications are appropriately protected.

353 Further assurance activities are associated with the specific protocols.

### 2.7.2 FTP_TRP.1 Trusted Path

#### 2.7.2.1 TSS

354 The evaluator shall examine the TSS to determine that the methods of remote TOE administration are indicated, along with how those communications are protected. The evaluator shall also confirm that all protocols listed in the TSS in support of TOE administration are consistent with those specified in the requirement, and are included in the requirements in the ST.

#### 2.7.2.2 Guidance Documentation

355 The evaluator shall confirm that the guidance documentation contains instructions for establishing the remote administrative sessions for each supported method.

#### 2.7.2.3 Tests

356 The evaluator shall perform the following tests:

a) Test 1: The evaluators shall ensure that communications using each specified (in the guidance documentation) remote administration method is tested during the course of the evaluation, setting up the connections as described in the guidance documentation and ensuring that communication is successful.

b) Test 2: For each protocol that the TOE can initiate as defined in the requirement, the evaluator shall follow the guidance documentation to ensure that in fact the communication channel can be initiated from the TOE.

c) Test 3: The evaluator shall ensure, for each communication channel with an authorized IT entity, the channel data is not sent in plaintext.

d) Test 4: The evaluators shall ensure that, for each protocol associated with each authorized IT entity tested during test 1, the connection is

physically interrupted. The evaluator shall ensure that when physical connectivity is restored, communications are appropriately protected.

357 Further assurance activities are associated with the specific protocols.

# 3 Evaluation Activities for SARs

358 The sections below specify Evaluation Activities for the Security Assurance Requirements included in the related cPPs (see section 1.1 above). The Evaluation Activities are an interpretation of the more general CEM assurance requirements as they apply to the specific technology area of the TOE.

359 In cases where the requirements are not technology dependent, the evaluator is expected to perform the CEM work units (e.g., ASE (except as in section 3.1), ALC_CMC.1, ALC_CMS.1), those activities are not repeated here, rather they are expressed as part of the cPP.

## 3.1 ASE: Security Target Evaluation

360 An evaluation activity is defined here for evaluation of Exact Conformance claims against a cPP in a Security Target. Other aspects of ASE remain as defined in [CEM, 10].

### 3.1.1 Conformance claims (ASE_CCL.1)

361 The table below indicates the actions to be taken for particular ASE_CCL.1 elements in order to determine exact conformance with a cPP.

| ASE_CCL.1 element | Evaluator Action |
|---|---|
| ASE_CCL.1.8C | The evaluator shall check that the statements of security problem definition in the PP and ST are identical. |
| ASE_CCL.1.9C | The evaluator shall check that the statements of security objectives in the PP and ST are identical. |
| ASE_CCL.1.10C | The evaluator shall check that the statements of security requirements in the ST include all the mandatory SFRs in the cPP, and all of the selection-based SFRs that are entailed by selections made in other SFRs (including any SFR iterations added in the ST). The evaluator shall check that if any other SFRs are present in the ST (apart from iterations of SFRs in the cPP) then these are taken only from the list of optional SFRs specified in the cPP (the cPP will not *necessarily* include optional SFRs, but may do so). If optional SFRs from the cPP are included in the ST then the evaluator shall check that any selection-based SFRs entailed by the optional SFRs adopted are also included in the ST. |

## 3.2 ADV: Development

### 3.2.1 Basic Functional Specification (ADV_FSP.1)

362 The Evaluation Activities for this assurance component focus on understanding the interfaces presented in the TOE Summary Specification (TSS) in response to the functional requirements, and on the interfaces presented in the AGD documentation. Specific requirements on this documentation are identified (where relevant) for each SFR in section 2 above, and in Evaluation Activities for AGD, ATE and AVA SARs in other parts of section 3 in this Supporting Document.

#### 3.2.1.1 Evaluation Activity:

363 The evaluator shall check the interface documentation to ensure it describes the purpose and method of use for each TSFI that is identified as being security relevant.

364 In this context, TSFI are deemed security relevant if they are used by the administrator to configure the TOE, or to perform other administrative functions (e.g., audit review or performing updates). Additionally, those interfaces that are identified in the ST, or guidance documentation, as adhering to the security policies (as presented in the SFRs), are also considered security relevant. The intent, is that these interfaces will be adequately tested, and having an understanding of how these interfaces are used in the TOE is necessary to ensure proper test coverage is applied.

365 The evaluator shall check the interface documentation to ensure it identifies and describes the parameters for each TSFI that is identified as being security relevant.

366 The documents to be examined for this assurance component in an evaluation are therefore the Security Target, AGD documentation, and any supplementary information required by the cPP for aspects such as entropy analysis or cryptographic key management architecture[1]: no additional "functional specification" documentation is necessary to satisfy the Evaluation Activities. The interfaces that need to be evaluated are also identified by reference to the assurance activities listed for each SFR, and are expected to be identified in the context of the Security Target, AGD documentation, and any supplementary information required by the cPP rather than as a separate list specifically for the purposes of CC evaluation. The direct identification of documentation requirements and their assessment as part of the Evaluation Activities for each SFR also means that the tracing

[1] The Security Target and AGD documentation are public documents. Supplementary information may be public or proprietary: the cPP and/or Evaluation Activity descriptions will identify where such supplementary documentation is permitted to be proprietary and non-public.

required in ADV_FSP.1.2D is treated as implicit, and no separate mapping information is required for this element.

367 However, if the evaluator is unable to perform some other required Evaluation Activity because there is insufficient design and interface information, then the evaluator is entitled to conclude that an adequate functional specification has not been provided, and hence that the verdict for the ADV_FSP.1 assurance component is a 'fail'.

## 3.3 AGD: Guidance Documents

368 It is not necessary for a TOE to provide separate documentation to meet the individual requirements of AGD_OPE and AGD_PRE. Although the Evaluation Activities in this section are described under the traditionally separate AGD families, the mapping between real TOE documents and AGD_OPE and AGD_PRE requirements may be many-to-many, as long as all requirements are met in documentation that is delivered to administrators and users (as appropriate) as part of the TOE.

### 3.3.1 Operational User Guidance (AGD_OPE.1)

369 Specific requirements and checks on the user guidance documentation are identified (where relevant) in the individual Evaluation Activities for each SFR, and for some other SARs (e.g. ALC_CMC.1).

#### 3.3.1.1 Evaluation Activity:

370 The evaluator shall check the requirements below are met by the guidance documentation.

371 Guidance documentation shall be distributed to administrators and users (as appropriate) as part of the TOE, so that there is a reasonable guarantee that administrators and users are aware of the existence and role of the documentation in establishing and maintaining the evaluated configuration.

372 Guidance documentation must be provided for every Operational Environment that the product supports as claimed in the Security Target and must adequately address all platforms claimed for the TOE in the Security Target.

373 The contents of the guidance documentation will be verified by the Evaluation Activities defined below and as appropriate for each individual SFR in section 2 above.

374 In addition to SFR-related Evaluation Activities, the following information is also required.

a) The guidance documentation shall contain instructions for configuring any cryptographic engine associated with the evaluated configuration of the TOE. It shall provide a warning to the

administrator that use of other cryptographic engines was not evaluated nor tested during the CC evaluation of the TOE.

b) The documentation must describe the process for verifying updates to the TOE by verifying a digital signature. The evaluator shall verify that this process includes the following steps:

1) Instructions for obtaining the update itself. This should include instructions for making the update accessible to the TOE (e.g., placement in a specific directory).

2) Instructions for initiating the update process, as well as discerning whether the process was successful or unsuccessful. This includes generation of the hash/digital signature.

c) The TOE will likely contain security functionality that does not fall in the scope of evaluation under this cPP. The guidance documentation shall make it clear to an administrator which security functionality is covered by the Evaluation Activities.

### 3.3.2 Preparative Procedures (AGD_PRE.1)

375 As for the guidance documentation, specific requirements and checks on the preparative procedures are identified (where relevant) in the individual Evaluation Activities for each SFR.

#### 3.3.2.1 Evaluation Activity:

376 The evaluator shall check the requirements below are met by the preparative procedures.

377 The contents of the preparative procedures will be verified by the Evaluation Activities defined below and as appropriate for each individual SFR in section 2 above.

378 Preparative procedures shall be distributed to administrators and users (as appropriate) as part of the TOE, so that there is a reasonable guarantee that administrators and users are aware of the existence and role of the documentation in establishing and maintaining the evaluated configuration.

379 The contents of the preparative procedures will be verified by the Evaluation Activities defined below and as appropriate for each individual SFR in section 2 above.

380 In addition to SFR-related Evaluation Activities, the following information is also required.

381 Preparative procedures must include a description of how the adminstrator verifies that the operational environment can fulfil its role to support the security functionality (including the requirements of the Security Objectives for the Operational Environment specified in the Security Target). The

documentation should be in an informal style and should be written with sufficient detail and explanation that they can be understood and used by the target audience (which will typically include IT staff who have general IT experience but not necessarily experience with the TOE product itself).

382 Preparative procedures must be provided for every Operational Environment that the product supports as claimed in the Security Target and must adequately address all platforms claimed for the TOE in the Security Target.

383 The preparative procedures must include

a) instructions to successfully install the TSF in each Operational Environment; and

b) instructions to manage the security of the TSF as a product and as a component of the larger operational environment; and

c) instructions to provide a protected administrative capability.

## 3.4 ATE: Tests

### 3.4.1 Independent Testing – Conformance (ATE_IND.1)

384 Testing is performed to confirm the functionality described in the TSS as well as the guidance documentation. The focus of the testing is to confirm that the requirements specified in the SFRs are being met.

385 The evaluator should consult Appendix B when determining the appropriate strategy for testing multiple variations or models of the TOE that may be under evaluation.

386 The SFR-related Evaluation Activities in the SD identify the specific testing activities necessary to verify compliance with the SFRs. The tests identified in these other Evaluation Activities constitute a sufficient set of tests for the purposes of meeting ATE_IND.1.2E. It is important to note that while the Evaluation Activities identify the testing that is necessary to be performed, the evaluator is responsible for ensuring that the interfaces are adequately tested for the security functionality specified for each SFR.

#### 3.4.1.1 Evaluation Activity:

387 The evaluator shall examine the TOE to determine that the test configuration is consistent with the configuration under evaluation as specified in the ST.

388 The evaluator shall examine the TOE to determine that it has been installed properly and is in a known state.

389 The evaluator shall prepare a test plan that covers all of the testing actions for ATE_IND.1 in the CEM and in the SFR-related Evaluation Activities. While it is not necessary to have one test case per test listed in an Evaluation Activity, the evaluator must show in the test plan that each applicable testing requirement in the SFR-related Evaluation Activities is covered.

390 The test plan identifies the platforms to be tested, and for any platforms not included in the test plan but included in the ST, the test plan provides a justification for not testing the platforms. This justification must address the differences between the tested platforms and the untested platforms, and make an argument that the differences do not affect the testing to be performed. It is not sufficient to merely assert that the differences have no affect; rationale must be provided. If all platforms claimed in the ST are tested, then no rationale is necessary.

391 The test plan describes the composition and configuration of each platform to be tested, and any setup actions that are necessary beyond what is contained in the AGD documentation. It should be noted that the evaluator is expected to follow the AGD documentation for installation and setup of each platform either as part of a test or as a standard pre-test condition. This may include special test drivers or tools. For each driver or tool, an argument (not just an assertion) should be provided that the driver or tool will not adversely affect the performance of the functionality by the TOE and its platform. This also includes the configuration of any cryptographic engine to be used (e.g. for cryptographic protocols being evaluated).

392 The test plan identifies high-level test objectives as well as the test procedures to be followed to achieve those objectives, and the expected results.

393 The test report (which could just be an updated version of the test plan) details the activities that took place when the test procedures were executed, and includes the actual results of the tests. This shall be a cumulative account, so if there was a test run that resulted in a failure, so that a fix was then installed and then a successful re-run of the test was carried out, then the report would show a "fail" result followed by a "pass" result (and the supporting details), and not just the "pass" result[2].

## 3.5 AVA: Vulnerability Assessment

### 3.5.1 Vulnerability Survey (AVA_VAN.1)

#### 3.5.1.1 Evaluation Activity:

394 The evaluator shall document their analysis and testing of potential vulnerabilities with respect to this requirement. This report could be included as part of the test report for ATE_IND, or could be a separate document.

395 The evaluator formulates hypotheses in accordance with process defined in Appendix A. The evaluator documents the flaw hypotheses generated for the TOE in the report in accordance with the guidelines in Appendix A.5. The evaluator shall then perform vulnerability analysis in accordance with

[2] It is not necessary to capture failures that were due to errors on the part of the tester or test environment. The intention here is to make absolutely clear when a planned test resulted in a change being required to the originally specified test configuration in the test plan, to the evaluated configuration identified in the ST and guidance documentation, or to the TOE itself.

Appendix A.4. The results of the analysis shall be documented in the report according to Appendix A.5.

# 4 Required Supplementary Information

396 This Supporting Document refers in various places to the possibility that 'supplementary information' may need to be supplied as part of the deliverables for an evaluation. This term is intended to describe information that is not necessarily included in the Security Target or guidance documentation, and that may not necessarily be public. Examples of such information could be entropy analysis, or description of a cryptographic key management architecture used in (or in support of) the TOE. The requirement for any such supplementary information will be identified in the relevant cPP.

397 The cPPs associated with this SD require an entropy analysis as described in [NDcPP] Appendix D.

# 5 References


[CC1] Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and General Model
CCMB-2012-09-001, Version 3.1 Revision 4, September 2012

[CC2] Common Criteria for Information Technology Security Evaluation,
Part 2: Security Functional Components,
CCMB-2012-09-002, Version 3.1 Revision 4, September 2012

[CC3] Common Criteria for Information Technology Security Evaluation,
Part 3: Security Assurance Components,
CCMB-2012-09-003, Version 3.1 Revision 4, September 2012

[CEM] Common Methodology for Information Technology Security Evaluation,
Evaluation Methodology,
CCMB-2012-09-004, Version 3.1, Revision 4, September 2012

[FWcPP] collaborative Protection Profile for Stateful Traffic Filter Firewalls,
Version 1.0, 27 February 2015

[NDcPP] collaborative Protection Profile for Network Devices,
Version 1.0, 27 February 2015

[VAWP] Draft cPP Vulnerability Analysis Whitepaper
Version 0.2 Draft

# A. Vulnerability Analysis

## A.1 Introduction

398 As noted in [VAWP], while vulnerability analysis is inherently a subjective activity, a minimum level of analysis can be defined and some measure of objectivity and repeatability (or at least comparability) can be imposed on the vulnerability analysis process. In order to achieve such objectivity and repeatability it is important that the evaluator follows a set of well-defined activities and documents his findings such that others can follow his arguments and come to the same conclusion as the evaluator in his report. While this does not guarantee that different evaluation facilities will identify exactly the same type of vulnerabilities or come to exactly the same conclusions, the approach defines the minimum level of analysis and the scope of that analysis, and provides schemes a measure of assurance that that minimum level of analysis is being performed by the evaluation facilities.

399 This supplemental guidance provides the information described in [VAWP] for the Network Device cPP, with modifications specific to this technology type.

400 The following section, "Additional Documentation", contains an iTC-developed list of documentation that is to be provided as input for the vulnerability assessment activities. This list of documentation is in addition to (but may be partially or fully duplicated by) documentation mandated by other SARs or evaluation activities.

401 The overall process follows that described in the [CEM] using the flaw-hypothesis methodology: evaluators formulate a list of potential flaws (the flaw hypotheses); evaluators investigate the flaws and disposition them; and evaluators write a report detailing their investigations. The following sections correspond to each of these activites. Section A.3 details the process that evaluators will follow to generate flaw hypotheses in each of four categories described in [VAWP]. Section A.4 describes the process the evaluators follow in dispositioning the flaws. Section A.5 describes the reporting aspects for the process, including the key details about what is and is not publically stated about the vulnerability assessment activity. Sections A.6 and A.7 contain information pertaining to flaw hypotheses that evaluators need to address that are generated by the iTC.

## A.2 Additional Documentation

402 [VAWP] indicates that the iTC determines appropriate additional documentation, based on the technology type, that will be made available to the evaluation team by the TOE developer. This documentation is in addition to that called out in the cPP evaluation activities and other SARs.

403 For the ND cPP, the additional documentation will at a minimum include the list of software and hardware components that comprise the TOE. Hardware components apply to all systems claimed in the ST, and should identify at a

minimum the network hardware and processors used by the TOE. Software components include the underlying operating environment/operating system, plus major components such as a web server, libraries such as protocol or cryptographic libraries, etc. This additional documentation is merely a list of the name and version number of the components, and will be used by the evaluators in formulating hypotheses during their analysis.

## A.3 Sources of vulnerability information

404 The method to be used in the vulnerability analysis for cPPs as outlined in [VAWP] is based on the flaw hypothesis methodology, where the evaluation team hypothesizes flaws and then either proves or disproves those flaws. Flaws are drawn from four sources (types):

a) A list of flaw hypotheses applicable to the technology described by the cPP (in this case, a network device) derived from Common Vulnerability Enumeration (CVE) or similar sources—there is a fixed set in the cPP/supplemental guidance that are agreed to by the iTC. Additionally, this will be supplemented with CVEs or entries for those same sources (as indicated below) that are directly applicable to the TOE or its identified components. The evaluators will also include in their assessment applicable entries in those sources CVEs that have been issued since the cPP was published;
b) A list of flaw hypotheses listed in the cPP/supplemental guidance that are derived from lessons learned specific to that technology and other iTC input (that might be derived from other open sources and vulnerability databases, for example); and
c) A list of flaw hypotheses derived from information available to the evaluators based on the SFRs and the baseline evidence provided by the vendor described in the cPP/supplemental guidance (includes section A.2), also including referenced public resources.
d) A list of flaw hypotheses that are generated through the use of TC-defined tools (e.g., nmap, fuzz testers) and their application may also be included.

### A.3.1 Type 1 Hypotheses – Public Vulnerability Database-Based

405 Section A.6 contains the list of public vulnerability sources, and entries in those sources to be considered for flaw hypotheses of type 1 above. In order to supplement this list, the evaluators shall also perform a search on those sources in Section A.6 that are more recent that the publication date of the cPP, and those that are specific to the TOE and its components as specified by the additional documentation mentioned above. Any duplicates – either in a specific entry, or the flaw hypothesis that is generated from an entry from the same or a different source – can be noted and removed from consideration by the evaluation team. It should be noted that the list in A.6 applies to all TOEs and indicate a type of flaw that is to be considered, which the evaluators use the search criteria (identified below) to collect the list of entries that apply *specifically* to the TOE. See the [VAWP] appendix for the analysis associated with CVE entries as an example.

406 The search criteria to be used when searching the sources published after the publication date of the cPP shall include:

- The terms “router” and “switch”
- The following protocols: TCP
- Any protocols not listed above supported (through an SFR) by the TOE (these will include at least one of the remote management protocols (IPsec, TLS, SSH))

407 As part of type 1 flaw hypothesis generation for the specific components of the TOE, the evaluator shall also search the component manufacturer’s websites to determine if flaw hypotheses can be generated on this basis (for instance, if security patches have been released for the version of the component being evaluated, the subject of those patches may form the basis for a flaw hypothesis).

### A.3.2 Type 2 Hypotheses – iTC Generated

408 Section A.7 contains the list of flaw hypothesis generated by the iTC for this cPP. Should the evaluators discover a type 3 or type 4 flaw that they believe should be considered in future versions of this cPP for inclusion in Section A.7, they will submit a non-proprietary write-up of the flaw for consideration by the iTC.

### A.3.3 Type 3 Hypotheses – Evaluation Team Generated

409 With respect to type 3 flaws, the evaluator is free to formulate flaws that are based on information presented by the product (through on-line help, product documentation and user guides, etc.) and product behaviour during the (functional) testing activities. The evaluator is also free to formulate flaws that are based on material that is not part of the baseline evidence (e.g., information gleaned from an Internet mailing list, or reading interface documentation on interfaces not included in the set provided by the developer), although such activities have the potential to vary significantly based upon the product and evaluation facility performing the analysis.

### A.3.4 Type 4 Hypotheses – Tool Generated

410 The evaluator shall perform the following activities to generate type 4 flaw hypotheses:

- Fuzz testing
    - Examine effects of sending:
        - mutated packets carrying each ‘Type’ and ‘Code’ value that is undefined in the relevant RFC for each of ICMPv4 (RFC 792) and ICMPv6 (RFC 4443)
        - mutated packets carrying each ‘Transport Layer Protocol’ value that is undefined in the respective RFC for IPv4

(RFC 791) IPv6 (RFC 2460) should also be covered if it is supported and claimed by the TOE.

Since none of these packets will belong to an allowed session, the packets should not be processed by the TOE, and the TOE should not be adversely affected by this traffic. Any results that are unexpected (e.g., core dumps) are candidates for a flaw hypothesis.

- Mutation fuzz testing of the remaining fields in the required protocol headers. This testing requires sending mutations of well-formed packets that have both carefully chosen and random values inserted into each header field in turn (i.e. testing is to include both carefully chosen and random insertion test cases). The original well-formed packets would be accepted as part of a normal existing communication stream and may still be accepted as valid packets when subject to the carefully chosen mutations (the individual packet alone would be valid although its contents may not be valid in the context of preceding and/or following packets), but will often not be valid packets when random values are inserted into fields. The carefully chosen values should include semantically significant values that can be determined from the type of the data that the field represents, such as values indicating positive and negative integers, boundary conditions, invalid binary combinations (e.g. for flag sets with dependencies between bits), and missing start or end values. Randomly chosen values may not result in well-formed packets, but are included nonetheless to see whether they can lead to the device entering an insecure state. Any results that are unexpected (e.g., core dumps) are candidates for a flaw hypothesis.

411 The iTC has not identified a specific tool to be used in accomplishing the above flaw hypothesis generation activity, so any tool used by the evaluation team is acceptable. The evaluation team shall record in the test report the name, version, parameters, and results of all test tools used for this this activity.

## A.4 Process for Evaluator Vulnerability Analysis

412 As flaw hypotheses are generated from the activities described above, the evaluation team will disposition them; that is, attempt to prove, disprove, or determine the non-applicability of the hypotheses. This process, as outlined in the [VAWP], is as follows.

413 The evaluator will refine each flaw hypothesis for the TOE and attempt to disprove it using the information provided by the developer or through penetration testing. During this process, the evaluator is free to interact with the developer without consulting the Certification Body (CB) to determine if

the flaw exists, including requests to the developer for additional evidence (e.g., detailed design information, consultation with engineering staff); however, the CB should be copied on all of these requests. Should the developer object to the information being requested as being not compatible with the overall level of the evaluation activity/cPP and cannot provide evidence otherwise that the flaw is disproved, the evaluator prepares an appropriate set of materials as follows: the source documents used in formulating the hypothesis, and why it represents a potential compromise against a specific TOE function; an argument why the flaw hypothesis could not be proven or disproved by the evidence provided so far; and the type of information required to investigate the flaw hypothesis further. The CB will then either approve or disapprove the request for additional information. If approved, the developer provides the requested evidence to disprove the flaw hypothesis (or, of course, acknowledge the flaw).

414 For each hypothesis, the evaluator will note whether the flaw hypothesis has been successfully disproved, successfully proven to have identified a flaw, or requires further investigation to be performed as part of the penetration testing effort. Again this can be dealt with in terms of meetings or written charts. It is important to have the results documented.

415 Should a flaw be found (either through the developer agreeing with the documentation analysis, or through the penetration effort), the evaluator will report these flaws to the vendor. All confirmed flaws should be addressed by the developer, and the resolution should be agreed to by the evaluator and noted as part of the evaluation report.

416 As indicated in Section A.5, the public statement with respect to vulnerability analysis that is performed on TOEs conformant to the cPP is constrained to coverage of flaws associated with Types 1 and 2 (defined in Section A.3) flaw hypotheses only. The fact that the iTC generates these candidate hypotheses indicates these must be addressed because they are exploitable by an attacker with Basic Attack Potential.

417 For flaws of Types 3 and 4, each CB will be responsible for determining what constitutes Basic Attack Potential for the purposes of determine whether a flaw is exploited in the TOE's environment. As this is a per-CB activity, no public claims are made with respect to the resistance of a particular TOE against flaws of Types 3 and 4; rather, the claim is that the activities outlined in this appendix were carried out, and the evaluation team and CB agreed that any residual vulnerabilities are not exploitable by an attacker with Basic Attack Potential.

## A.5 Reporting

418 The evaluators shall produce two reports on the testing effort; one that is public-facing (that is, included in the non-proprietary evaluation report) and one that is delivered to the overseeing CB.

419 The public-facing report is just a statement that the evaluators have examined the CVEs applicable to the product (Section A.6 plus additional CVE-based

hypotheses generated by the evaluators per Section A.3.1) and those specified in the cPP by the iTC and contained in Section A.7 (this encompasses hypotheses of Types 1 and 2 mentioned above). Additionally, there is a statement that the evaluation team developed Types 3 and 4 flaw hypotheses in accordance with Section A.3, and that no residual vulnerabilities exist that are exploitable by attackers with Basic Attack Potential as defined by the CB in accordance with the guidance in the CEM. No other information is provided in the public-facing report.

420 For the (internal) CB report, we suggest that the evaluation team must report all of the flaw hypotheses generated; all documentation used to generate the flaw hypotheses; and how each flaw hypothesis was resolved (this includes whether the original flaw hypothesis was confirmed or disproved, and any analysis relating to whether a residual vulnerability is exploitable by an attacker with Basic Attack Potential). In identifying the documentation used in coming up with the flaw hypotheses, the evaluation team must characterize the documentation so that a reader can determine whether it is strictly required by the support documents/evaluation activities (that is, it forms part of the baseline evidence), and the nature of the documentation (design information, developer engineering notebooks, etc.). At the conclusion of the evaluation, a set of interested CBs (subject to negotiation between all parties concerned) and perhaps other members of the iTC may review this information and make a determination of the impacts to supporting documents for future evaluations against that cPP (for example, if a large number of the flaw hypotheses were generated based on a certain type of documentation, then additional documentation in this area may be required by the iTC for future evaluations).

## A.6 Public Vulnerability Database Entries for Flaw Hypotheses

421 No entries are currently defined for this list

## A.7 Additional Flaw Hypotheses

422 No entries are currently defined for this list

## A.8 iTC Activities – cPP and Supporting Document Maintenance

423 As indicated in the preceding sections, the iTC plays a key role in determining the scope of the vulnerability analysis with respect to what is publically reported. This section details the activities that the iTC must perform in order to ensure that flaws to be investigated by the evaluation team are identified, are compatible with the overall level of assurance of a TOE conformant to this PP, and cover the areas of concern by the iTC for this technology.

424 There are four activities (and associated outputs) that need to accomplished by the iTC:

1) The iTC must determine what additional documentation is necessary for the evaluation team to examine in the course of their vulnerability

assessment. The cPP and associated Supporting Document detail a set of information that must be available (for example, information called for by the TSS in the various evaluation activities in the Supporting Document). If the iTC feels that additional documentation not already covered (or only partially covered) in the cPP or the Supporting Document is necessary, they define that information and include it in section A.2.

2) The iTC must determine what public vulnerability databases are to be used as the basis for Type 1 (Section A.3) hypotheses, and what entries in these databases apply. In performing this activities, the iTC first agrees upon the sources to be used; for instance, the Common Vulnerability Enumeration (CVE) list. Note that it is not expected that this be an exhaustive list; only that it is a list that the iTC feels gives good representation for vulnerabilities with respect to the technology type.

   Having identified the sources, for each source the iTC defines criteria for selecting entries in the list. The lists and criteria should be identified in section A.3.1 so that evaluators can use the same sources and criteria at evaluation time to select entries that were made after the cPP was published. For each entry that meets the criteria, the iTC determines whether or not to include it in the list to be considered by the evaluation team. This will likely necessitate the creation of some criteria by which to judge an entry that is agreed to by the iTC. For instance, CVEs that would generate flaw hypotheses related to buffer overflows would probably be rejected as a generic flaw hypothesis.

   The output of this activity would be a list of entries that the evaluation team would consider as applicable in generating flaw hypotheses. See the appendix to the [VAWP] for an example of this analysis activity as applied to firewalls using the CVE database.

3) The iTC must consider if there are any technology-specific vulnerabilities or types of vulnerabilities that the evaluators should consider that are not contained in section A.6. This could be based on previous evaluations against the cPP, experience of the iTC members, or other factors. This set of vulnerabilities (Type 2) would be captured in Section A.7 and would then need to be considered by the evaluation team. It is likely that there will be few or no entries identified for this type until more experience is gained with the cPP.

4) The final activity the iTC must perform is the identification of any tools or testing that would indicate the creation of flaw hypotheses (Type 4). The iTC can choose to merely outline the testing that needs to be formed, and/or they can identify specific tools and testing to be done with those tools. In this definition, the iTC also indicates test results that indicate that a flaw hypothesis should be made (the goal of this section is not to perform or re-do functional testing; it is to test in a way that might produce anomalies that are to be candidate flaw hypotheses). The iTC documents and specific tools; the procedures, settings, and

testing to be performed; and criteria for creating flaw hypotheses from these results in section A.3.4.

# B. Network Device Equivalency Considerations

## B.1 Introduction

425 This appendix provides a foundation for evaluators to determine whether a vendor's request for equivalency of products for different models wishing to claim conformance to the Network Device collaborative Protection Profiles. Separate TOE models may include differences that could necessitate separate testing across each model. If there are no variations in any of the categories listed below, the models may be considered equivalent.

426 Determination of equivalency between models can result in several different testing outcomes as described in section B.3.

427 If a set of TOE are determined to be equivalent, testing may be performed on a single variation of the TOE. However, if the TOE variations have security relevant functional differences, each of the TOE models that exhibits either functional or structural differences must be separately tested. Generally speaking, only the difference between each variation of TOE must be separately tested. Other equivalent functionality may be tested on a representative model and not across multiple platforms.

428 If a vendor disagrees with the evaluator's assessment of equivalency, the Certification Body arbitrates between the two parties whether equivalency exists.

## B.2 Evaluator guidance for determining equivalence

429 The following table provides a description of how an evaluator should consider each of the factors that affect equivalency between TOE model variations and across operating environments. Additionally, the table also identifies scenarios that will result in additional separate testing across models.

| Factor | Same/Not Same | Evaluator guidance |
|---|---|---|
| **Platform/Hardware Dependencies** | Independent | If there are no identified platform/hardware dependencies, the evaluator shall consider testing on multiple hardware platforms to be equivalent. |
| | Dependencies | If there are specified differences between platforms/hardware, the evaluator must identify if the differences affect the cPP-specified security functionality or if they apply to non-cPP-specified functionality. If functionality specified in the cPP is dependent upon platform/hardware provided services, the product must be tested on each of the different platforms to be considered |

| Factor | Same/Not Same | Evaluator guidance |
|---|---|---|
| | | validated on that particular hardware combination. In these cases, the evaluator has the option of only re-testing the functionality dependent upon the platform/hardware provided functionality. If the differences only affect non-cPP-specified functionality, the variations may still be considered equivalent. For each difference the evaluator must provide an explanation of why the difference does or does not affect cPP-specified functionality. |
| **Differences in TOE Software Binaries** | Identical | If the model binaries are identical, the model variations shall be considered equivalent. |
| | Different | If there are differences between model software binaries, a determination must be made if the differences affect cPP-specified security functionality. If cPP-specified functionality is affected, the models are not considered equivalent and must be tested separately. The evaluator has the option of only retesting the functionality that was affected by the software differences. If the differences only affect non-PP specified functionality, the models may still be considered equivalent. For each difference the evaluator must provide an explanation of why the difference does or does not affect cPP specified functionality. |
| **Differences in Libraries Used to Provide TOE Functionality** | Same | If there are no differences between the libraries used in various TOE models, the model variations shall be considered equivalent. |
| | Different | If the separate libraries are used between model variations, a determination of whether the functionality provided by the library affects cPP-specified functionality must be made. If cPP-specified functionality is affected, the models are not considered equivalent and must be tested separately. The evaluator has the option of only retesting the functionality that was affected by the differences in the included libraries. If the different libraries only affect non-PP specified functionality, the models may still be considered equivalent. For each different library, the evaluator must provide an explanation of why the different libraries do or do not affect cPP specified functionality. |
| **TOE Management Interface** | Consistent | If there are no differences in the management interfaces between various TOE models, the models |

| Factor | Same/Not Same | Evaluator guidance |
|---|---|---|
| **Differences** | | variations shall be considered equivalent. |
| | Differences | If the product provides separate interfaces based on the model variation, a determination must be made of whether cPP-specified functionality can be configured by the different interfaces. If the interface differences affect cPP-specified functionality, the variations are not considered equivalent and must be separately tested. The evaluator has the option of only retesting the functionality that can be configured by the different interfaces (and the configuration of said functionality). If the different management interfaces only affect non-PP specified functionality, the models may still be considered equivalent. For each management interface difference, the evaluator must provide an explanation of why the different management interfaces do or do not affect cPP specified functionality. |
| **TOE Functional Differences** | Identical | If the functionality provided by different TOE model variation is identical, the models variations shall be considered equivalent. |
| | Different | If the functionality provided by different TOE model variations differ, a determination must be made if the functional differences affect cPP-specified functionality. If cPP-specific functionality differs between models, the models are not considered equivalent and must be tested separately. In these cases, the evaluator has the option of only retesting the functionality that differs model-to-model. If the functional differences only affect non-cPP specified functionality, the model variations may still be considered equivalent. For each difference the evaluator must provide an explanation of why the difference does or does not affect cPP specified functionality. |

**Table 1 - Evaluation Equivalency Analysis**

## B.3 Strategy

430 When performing the equivalency analysis, the evaluator should consider each factor independently. Each analysis of an individual factor will result in one of three outcomes,

- For the particular factor, all variations of the TOE on all supported platforms are equivalent. In this case, testing may be performed on a

single model in a single test environment and cover all supported models and environments.

- For the particular factor, a subset of the product has been identified to require separate testing to ensure that it operates identically to all other equivalent TOE. The analysis would identify the specific combinations of models/testing environments that needed to be tested.
- For the particular factor, none of the variations of the TOE are equivalent and testing is therefore performed on all models and environments.

431 Complete CC testing of the product would encompass the totality of each individual analysis performed for each of the identified factors.

## B.4 Test presentation/Truth in advertising

432 In addition to determining what to test, the evaluation results and resulting Certification Report, must identify the actual module and testing environment combinations that have been tested. The analysis used to determine the testing subset may be considered proprietary and will only optionally be publicly included.