ASSURANCE ACTIVITY REPORT

ORACLE LINUX 9.3

PREPARED BY
EWA-Canada, An Intertek Company

PREPARED FOR
Canadian Centre for Cyber Security (CCCS) and
National Information Assurance Partnership (NIAP)

REPORT NO
2278-002-D007

DOCUMENT VERSION
0.8

DATE
14 February 2025

REV.	DATE	REVISION DETAILS
0.1	18 April 2024	PETR ASE Submission.
0.2	22 May 2024	Address certifier comments.
0.3	5 June 2024	Address certifier comments.
0.4	7 August 2024	PETR ADV/ENT Submission.
0.5	21 October 2024	Address certifier comments.
0.6	18 November 2024	Update CAVP certificate number.
0.7	02 February 2025	Update for TD0873.
0.8	14 February 2025	Address certifier comments.

CONTENTS

TOE Developer:

Oracle Corporation
500 Oracle Parkway
Redwood Shores, CA 94065
USA

Common Criteria Versions

- Common Criteria for Information Technology Security Evaluation Part 1: Introduction, Version 3.1, Revision 5, April 2017.
- Common Criteria for Information Technology Security Evaluation Part 2: Security Functional Components, Version 3.1, Revision 5, April 2017.
- Common Criteria for Information Technology Security Evaluation Part 3: Security Assurance Components, Version 3.1, Revision 5, April 2017.

Common Evaluation Methodology Versions

- Common Methodology for Information Technology Security Evaluation, Evaluation Methodology, Version 3.1, Revision 5, April 2017.

Protection Profiles

- Protection Profile for General Purpose Operating Systems, Version 4.3, 22 April 2019
- Functional Package for Secure Shell (SSH), Version 1.0, 13 May 2021
- Functional Package for Transport Layer Security (TLS), Version 1.1, 1 March 2019

NIAP Technical Decisions

ITEM	DETAILS
TD0873	Updating FIPS 186-4 to 186-5 in PP_OS_V4.3
TD0844	Addition of Assurance Package for Flaw Remediation V1.0 Conformance Claim
TD0839	Clarification for Local Administration in FTP_TRP.1.3
TD0821	Corrections to ECD for PP_OS_V4.3
TD0812	Updated CC Conformance Claims in PP_OS_V4.3
TD0789	Correction to TLS Selection in FIA_X509_EXT.2.1
TD0773	Updates to FIA_X509_EXT.1 for Exception Processing and Test Conditions
TD0713	Functional Package SFR mappings to objectives

TD0712	Support for Bluetooth Standard 5.3
TD0701	Incomplete selection reference in FCS_CKM_EXT.4 TSS activities
TD0696	Removal of 160 bit selection from FCS_COP.1/HASH & FCS_COP.1/KEYHMAC
TD0693	Typos in OSPP 4.3
TD0691	OSPP 4.3 Conditional authentication testing
TD0675	Make FPT_W^X_EXT.1 Optional

Table 1 - NIAP Technical Decisions for PP_OS_v4.3

ITEM	DETAILS
TD0779	Updated Session Resumption Support in TLS package V1.1
TD0770	TLSS.2 connection with no client cert
TD0739	PKG_TLS_V1.1 has 2 different publication dates
TD0726	Corrections to (D)TLSS SFRs in TLS 1.1 FP
TD0513	CA Certificate loading
TD0499	Testing with pinned certificates
TD0469	Modification of test activity for FCS_TLSS_EXT.1.1 test 4.1
TD0442	Updated TLS Ciphersuites for TLS Package

Table 2 - NIAP Technical Decisions for PKG_TLS_V1.1

ITEM	DETAILS
TD0777	Clarification to Selections for Auditable Events for FCS_SSH_EXT.1
TD0732	FCS_SSHS_EXT.1.3 Test 2 Update
TD0695	Choice of 128 or 256 bit size in AES-CTR in SSH Functional Package.
TD0682	Addressing Ambiguity in FCS_SSHS_EXT.1 Tests

Table 3 - NIAP Technical Decisions for PKG_SSH_V1.0

1. INTRODUCTION

This document presents assurance activity evaluation results of the TOE evaluation. There are three types of assurance activities, and the following is provided for each:

1. TOE Summary Specification (TSS) - An indication that the required information is in the TSS section of the Security Target.
2. Guidance - A specific reference to the location in the guidance is provided for the required information.
3. Test – A summary of the test procedure used, and the results obtained is provided for each required test activity.

This Assurance Activities Report contains sections for each functional class and family and sub-sections addressing each of the SFRs specified in the Security Target.

1.1 Evidence

The following is a list of the documents consulted:

- [ST] Security Target, v 1.5, 14 February 2025.
- [ENTROPY] Entropy Assessment Report, v 0.12, 16 September 2024.
- [AGD] Common Criteria Guidance Document, v 1.4, 14 February 2025.
- [PP_OS_v4.3] Protection Profile for General Purpose Operating Systems, Version 4.3, 27 September 2022.
- [PKG_SSH] Functional Package for Secure Shell (SSH), Version 1.0, 13 May 2021
- [PKG_TLS] Functional Package for Transport Layer Security (TLS), Version 1.1, 1 March 2019

2. Security Functional Requirement Assurance Activities

This section describes the assurance activities associated with the SFRs defined in the PP and the results of those activities as performed by the evaluation team. The assurance activities are extracted from [PP_OS_v4.3].

2.1 Cryptographic Support (FCS)

2.1.1 FCS_CKM.1 Cryptographic Key Generation (Refined)

2.1.1.1 TSS Assurance Activity [TD0712]

The evaluator will ensure that the TSS identifies the key sizes supported by the OS. If the ST specifies more than one scheme, the evaluator will examine the TSS to verify that it identifies the usage for each scheme. If "P-256" is selected, the evaluator will examine the TSS to verify that it is only used for Bluetooth functions.

Evaluator Assessment:

Table 19 in Section 7 of the [ST] identifies the following:

RSA keys (3072, and 4096 bits) for key generation conforming to FIPS PUB 186-5 (TLS and SSH communications)

ECC schemes using NIST curves P-384, and P-521 that meet the following: FIPS PUB 186-5 "Digital Signature Standard (DSS). ECDSA is used in support of TLS and SSH communications.

FFC (3072, and 4096) for key generation using safe primes as specified in NIST SP 800-56A Revision 3.

2.1.1.2 Guidance Documentation Assurance Activity

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

Evaluator Assessment:

Section 5 and Section 6 of the [AGD] provide guidance on how to configure the TOE to use the selected key generation schemes and key sizes for all uses defined in the PP.

2.1.1.3 Test Assurance Activity [TD0712]

Evaluation Activity Note: The following tests may require the vendor to furnish a developer environment and developer tools that are typically not available to end-users of the OS.

The following content should be included if:

- RSA schemes is selected from FCS_CKM.1.1

Key Generation for FIPS PUB 186-5 RSA Schemes
The evaluator will verify the implementation of RSA Key Generation by the OS 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. Key Pair generation specifies 5 ways (or methods) to generate the primes p and q. These include:

1. Random Primes:
    - Provable primes
    - Probable primes
2. 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

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 will verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated from a known good implementation.

If possible, the Random Probable primes method should also be verified against a known good implementation as described above. Otherwise, the evaluator will have the TSF generate 10 keys pairs for each supported key length nlen and verify:

- n = p ·q,
- p and q are probably prime according to Miller-Rabin tests,
- GCD(p-1,e) = 1,
- GCD(q-1,e) = 1,
- 216 = e = 2256 and e is an odd integer,
- |p-q| > 2nlen/2 - 100,
- p = 2nlen/2 -1/2,
- q = 2nlen/2 -1/2,
- 2(nlen/2) < d < LCM(p-1,q-1),
- e·d = 1 mod LCM(p-1,q-1).

The following content should be included if:

- ECC schemes using “NIST curves” is selected from FCS_CKM.1.1

Key Generation for Elliptic Curve Cryptography (ECC)
FIPS 186-5 ECC Key Generation Test

For each supported NIST curve, i.e., P-256, P-384 and P-521, the evaluator will 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 will submit the generated key pairs to the public key

verification (PKV) function of a known good implementation.

FIPS 186-5 Public Key Verification (PKV) Test

For each supported NIST curve, i.e., P-256, P-384 and P-521, the evaluator will 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 will obtain in response a set of 10 PASS/FAIL values.

The following content should be included if:

- FIPS PUB 186-5 is selected from FCS_CKM.1.1

Key Generation for Finite-Field Cryptography (FFC)
The evaluator will 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.

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

Cryptographic and Field Primes:

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

and two ways to generate the cryptographic group generator g:
Cryptographic Group Generator:

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

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

- 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

The security strength of the RBG must be at least that of the security offered by the FFC parameter set. 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. For each key length supported, the evaluator will have the TSF generate 25 parameter sets and key pairs. The evaluator will 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

- gq mod p = 1
- gx mod p = y

for each FFC parameter set and key pair.

Evaluator Assessment:

Testing is satisfied through the Cryptographic Algorithm Verification Program (CAVP). The cryptographic implementation used by the TOE received CAVP Certificate A6141.

2.1.2 FCS_CKM.2 Cryptographic Key Establishment (Refined)

2.1.2.1 TSS Assurance Activity

The evaluator will 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 will examine the TSS to verify that it identifies the usage for each scheme.

Evaluator Assessment:

Table 19 in Section 7 of the [ST] identifies the supported key establishment schemes. The schemes identified (ECC, FFC) correspond with the key generation schemes identified in FCS_CKM.1.1.

2.1.2.2 Guidance Documentation Assurance Activity

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

Evaluator Assessment:

Section 5 and Section 6 of the [AGD] provides guidance on how to configure the TOE to use the selected key establishment schemes.

2.1.2.3 Test Assurance Activity

Evaluation Activity 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 Establishment Schemes

The evaluator will verify the implementation of the key establishment schemes supported by the OS using the applicable tests below.

The following content should be included if:

- Elliptic curve-based, Finite field-based is selected from FCS_CKM.2.1

SP800-56A Key Establishment Schemes

The evaluator will verify the OS'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 the OS has implemented the components of the key agreement scheme according to the specifications in the Recommendation. These components include the calculation of the discrete logarithm cryptography (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 will 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 MAC data and the calculation of MAC tag.

Function Test

The Function test verifies the ability of the OS to implement the key agreement schemes correctly. To conduct this test the evaluator will generate or obtain test vectors from a known good implementation of the OS's 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 will generate 10 sets of test vectors. The data set consists of one set of domain parameter values (FCC) 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.

The evaluator will obtain the DKM, the corresponding OS'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 OS id fields.

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

The evaluator will 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.

If key confirmation is supported, the OS will perform the above for each implemented approved MAC algorithm.

Validity Test

The Validity test verifies the ability of the OS to recognize another party's valid and invalid key agreement results with or without key confirmation. To conduct this test, the evaluator will obtain a list of the supporting cryptographic functions included in the SP800-56A Revision 3 key agreement implementation to determine which errors the OS should be able to recognize. The evaluator generates a set of 24 FCC or 30 ECC test vectors consisting of data sets including domain parameter values or NIST approved curves, the evaluator's public keys, the OS's public/private key pairs, MAC tag, and any inputs used in the KDF, such as the other info and OS id fields.

The evaluator will inject an error in some of the test vectors to test that the OS 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 MAC tag. If the OS 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 OS's static private key to assure the OS 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 will remain unmodified and therefore should result in valid key agreement results (they should pass).

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

The following content should be included if:

- RSA-based, is selected from FCS_CKM.2.1

RSAES-PKCS1-v1_5 Key Establishment Schemes

The evaluator shall verify the correctness of the TSF's implementation of RSAES-PKCS1-v1_5 by using a known good implementation for each protocol selected in FTP_ITC_EXT.1 that uses RSAES-PKCS1-v1_5.

The following content should be included if:

- Finite field-based is selected from FCS_CKM.2.1

FFC Schemes using "safe-prime" groups (identified in Appendix D of SP 800-56A Revision 3)

The evaluator shall verify the correctness of the TSF's implementation of "safe-prime" groups by using a known good implementation for each protocol selected in FTP_ITC_EXT.1 that uses "safe-prime" groups. This test must be performed for each "safe-prime" group that each protocol uses.

Evaluator Assessment:

Testing is satisfied through the Cryptographic Algorithm Verification Program (CAVP). The cryptographic implementation used by the TOE received CAVP Certificate A6141.

2.1.3 FCS_CKM_EXT.4 Cryptographic Key Destruction

2.1.3.1 TSS Assurance Activity [TD0701]

The evaluator examines the TSS to ensure it describes how the keys are managed in volatile memory. This description includes details of how each identified key is introduced into volatile memory (e.g. by derivation from user input, or by unwrapping a wrapped key stored in non-volatile memory) and how they are overwritten.

The evaluator will check to ensure the TSS lists each type of key that is stored in in non-volatile memory, and identifies how the TOE interacts with the underlying platform to manage keys (e.g., store, retrieve, destroy). The description includes details on the method of how the TOE interacts with the platform, including an identification and description of the interfaces it uses to manage keys (e.g., file system APIs, platform key store APIs).

If the ST makes use of the open assignment and fills in the type of pattern that is used, the evaluator examines the TSS to ensure it describes how that pattern is obtained and used. The evaluator will verify that the pattern does not contain any CSPs.

The evaluator will check that the TSS identifies any configurations or circumstances that may not strictly conform to the key destruction requirement.

If the selection “destruction of all key encrypting keys (KEKs) protecting the target key according to FCS_CKM_EXT.4.1, where none of the KEKs protecting the target key are derived” is included the evaluator will examine the TOE’s keychain in the TSS and identify each instance when a key is destroyed by this method. In each instance the evaluator shall verify all keys capable of decrypting the target key are destroyed in accordance with a specified key destruction method in FCS_CKM_EXT.4.1 The evaluator shall verify that all of the keys capable of decrypting the target key are not able to be derived to re-establish the keychain after their destruction.

Evaluator Assessment:

Table 19 in Section 7 of the [ST] identifies the following:

For volatile memory, the destruction shall be executed by removal of power to the memory.

For non-volatile memory, the destruction consists of the invocation of an interface provided by the underlying platform that instructs the underlying platform to destroy the abstraction that represents the key.

Symmetric key material and Diffie-Hellman / EC Diffie-Hellman public and private keys are derived using the SSH KDF and stored in volatile memory.

Asymmetric key material is stored in non-volatile memory. The /etc/ssh directory contains the host keys which are generated using ssh-keygen. The $HOME/.ssh directory contains user keys and are generated using ssh-keygen. Authorized public keys are generated remotely and input into the TOE.

Symmetric session keys for TLS are derived from the TLS KDF.

The TSS does not identify any configurations or circumstances that may not strictly conform the key destruction requirement.

2.1.3.2 Guidance Documentation Assurance Activity

Operational Guidance

There are a variety of concerns that may prevent or delay key destruction in some cases. The evaluator will check that the guidance documentation identifies configurations or circumstances that may not strictly conform to the key destruction requirement, and that this description is consistent with the relevant parts of the TSS and any other relevant Required Supplementary Information. The evaluator will check that the guidance documentation provides guidance on situations where key destruction may be delayed at the physical layer and how such situations can be avoided or mitigated if possible.

Some examples of what is expected to be in the documentation are provided here.

When the TOE does not have full access to the physical memory, it is possible that the storage may be implementing wear-leveling and garbage collection. This may create additional copies of the key that are logically inaccessible but persist physically. In this case, to mitigate this the drive should support the TRIM command and implements garbage collection to destroy these persistent copies when not actively engaged in other tasks.

Drive vendors implement garbage collection in a variety of different ways, as such there is a variable amount of time until data is truly removed from these solutions. There is a risk that data may persist for a longer amount of time if it

is contained in a block with other data not ready for erasure. To reduce this risk, the operating system and file system of the OE should support TRIM, instructing the non-volatile memory to erase copies via garbage collection upon their deletion. If a RAID array is being used, only set-ups that support TRIM are utilized. If the drive is connected via PCI-Express, the operating system supports TRIM over that channel.

The drive should be healthy and contains minimal corrupted data and should be end-of-lifed before a significant amount of damage to drive health occurs, this minimizes the risk that small amounts of potentially recoverable data may remain in damaged areas of the drive.

Evaluator Assessment:

Section 7 of the [AGD] describes circumstances where SSDs levelling mechanism may prevent software from overwriting the exact physical location of the keys. Mitigation of this issue is also described in this section.

Keys held in volatile memory are destroyed by removal of power to the TOE.

2.1.3.3 Test Assurance Activity

Test 1: Applied to each key held as in volatile memory and subject to destruction by overwrite by the TOE (whether or not the value is subsequently encrypted for storage in volatile or non-volatile memory). In the case where the only selection made for the destruction method key was removal of power, then this test is unnecessary. The evaluator will:

1. Record the value of the key in the TOE subject to clearing.
2. Cause the TOE to perform a normal cryptographic processing with the key from Step #1.
3. Cause the TOE to clear the key.
4. Cause the TOE to stop the execution but not exit.
5. Cause the TOE to dump the entire memory of the TOE into a binary file.

6. Search the content of the binary file created in Step #5 for instances of the known key value from Step #1.

Steps 1-6 ensure that the complete key does not exist anywhere in volatile memory. If a copy is found, then the test fails

Evaluator Assessment:

N/A. The TOE selects removal of power as the key destruction method.

Test 2: Applied to each key held in non-volatile memory and subject to destruction by the TOE. The evaluator will use special tolls (as needed), provided by the TOE developer if necessary, to ensure the tests function as intended.

1. Identify the purpose of the key and what access should fail when it is deleted. (e.g. the data encryption key being deleted will cause data decryption to fail.)
2. Cause the TOE to clear the key.
3. Have the TOE attempt the functionality that the cleared key would be necessary for.

The test succeeds if step 3 fails.

Evaluator Assessment:

Test 2: The TOE is not able to use the keys after being cleared in non-volatile memory.

Test 3:

Tests 3 and 4 do not apply for the selection instructing the underlying platform to destroy the representation of the key, as the TOE has no visibility into the inner workings and completely relies on the underlying platform.

The following tests are used to determine the TOE is able to request the platform to overwrite the key with a TOE supplied pattern.

Applied to each key held in non-volatile memory and subject to destruction by overwrite by the TOE. The evaluator will use a tool that provides a logical view of the media (e.g., MBR file system):

1. Record the value of the key in the TOE subject to clearing
2. Cause the TOE to perform a normal cryptographic processing with the key from Step #1.
3. Cause the TOE to clear the key.
4. Search the logical view that the key was stored in for instances of the known key value from Step #1. If a copy is found, then the test fails.

Evaluator Assessment:

N/A. The TOE selects instructing the underlying platform to destroy the representation of the key.

Test 4: Applied to each key held as non-volatile memory and subject to destruction by overwrite by the TOE. The evaluator will use a tool that provides a logical view of the media:

1. Record the logical storage location of the key in the TOE subject to clearing.
2. Cause the TOE to perform a normal cryptographic processing with the key from Step #1.
3. Cause the TOE to clear the key.
4. Read the logical storage location in Step #1 of non-volatile memory to ensure the appropriate patter is utilized.

The test succeeds if the correct patter is used to overwrite the key in the memory location. If the patter is not found the test fails.

Evaluator Assessment:

N/A. The TOE selects instructing the underlying platform to destroy the representation of the key.

2.1.4 FCS_COP.1/ENCRYPT Cryptographic Operation – Encryption/Decryption (Refined)

2.1.4.1 TSS Assurance Activity [TD0712]

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

If "128-bit" is selected, the evaluator will examine the TSS to verify that 128-bit is only used with AES-CCM for Bluetooth functions.

Evaluator Assessment:

Table 19 in Section 7 of the [ST] identifies the key size supported as 256 bits. Usage of each AES scheme is described in the TSS.

2.1.4.2 Guidance Documentation Assurance Activity

The evaluator will verify that the AGD documents contains instructions required to configure the OS to use the required modes and key sizes.

Evaluator Assessment:

Section 5.1 – Configuring SSH server and Section 5.2 – Configuring SSH Client in the [AGD] document provides instructions to configure the required modes and key sizes.

The evaluator examined Section 6 - Configuring TLS in the [AGD] document and found that no key configuration was necessary for the TOE to use the required modes and key sizes.

2.1.4.3 Test Assurance Activity [TD0712]

The evaluator will execute all instructions as specified to configure the OS to the appropriate state. The evaluator will perform all of the following tests for each algorithm implemented by the OS and used to satisfy the requirements of this PP:

The following content should be included if:

- AES-XTS is selected from FCS_COP.1.1/ENCRYPT

XTS-AES Test

The evaluator will test the encrypt functionality of XTS-AES for each combination of the following input parameter lengths:

- 512 bit (for AES-256) key
- Three data unit (i.e., plaintext) lengths. One of the data unit lengths will be a nonzero integer multiple of 256 bits, if supported. One of the data unit lengths will be an integer multiple of 256 bits, if supported. The third data unit length will be either the longest supported data unit length or 216 bits, whichever is smaller.

using a set of 100 (key, plaintext and 256-bit random tweak value) 3-tuples and obtain the ciphertext that results from XTS-AES encrypt.

The evaluator may supply a data unit sequence number instead of the tweak value if the implementation supports it. The data unit sequence number is a base-10 number ranging between 0 and 255 that implementations convert to a tweak value internally.

The evaluator will test the decrypt functionality of XTS-AES using the same test as for encrypt, replacing plaintext values with ciphertext values and XTS-AES encrypt with XTSAES decrypt.

Evaluator Assessment:

N/A. XTS-AES is not claimed.

The following content should be included if:

- AES-CBC is selected from FCS_COP.1.1/ENCRYPT

AES-CBC Known Answer Tests

There are four Known Answer Tests (KATs), described below. In all KATs, the plaintext, ciphertext, and IV values will be 256-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 will compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

- Test 5: To test the encrypt functionality of AES-CBC, the evaluator will supply a set of 5 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. The plaintext values will encrypted with a 256-bit all-zeros key. To test the decrypt functionality of AES-CBC, the evaluator will perform the same test as for encrypt, using 5 ciphertext values as input and AES-CBC decryption.
- Test 6: To test the encrypt functionality of AES-CBC, the evaluator will supply a set of five 256-keys 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. To test the decrypt functionality of AES-CBC, the evaluator will perform the same test as for encrypt, using an all-zero ciphertext value as input and AES-CBC decryption.
- Test 7: To test the encrypt functionality of AES-CBC, the evaluator will supply the a 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. Key i will have the leftmost i bits be ones and the rightmost N-i bits be zeros, for i in [1,N]. To test the decrypt functionality of AES-CBC, the evaluator will supply the set 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 set of key/ciphertext pairs will have 256 256-bit key/ciphertext pairs. Key i in each set will 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 will be the value that results in an all-zeros plaintext when decrypted with its corresponding key.
- Test 8: To test the encrypt functionality of AES-CBC, the evaluator will supply the set of 256 plaintext values described below and obtain the ciphertext values that result from AES-CBC encryption of the given plaintext using a 256-bit key value of all zeros with an IV of all zeros. Plaintext value i in each set will have the leftmost i bits be ones and the rightmost 256-i bits be zeros, for i in [1,256].

To test the decrypt functionality of AES-CBC, the evaluator will 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

The evaluator will test the encrypt functionality by encrypting an i-block message where 1 < i ≤ 10. The evaluator will 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 will be compared to the result of encrypting the same plaintext message with the same key and IV using a known good implementation. The evaluator will also test the decrypt functionality for each mode by decrypting an i-block message where 1 < i ≤10. The evaluator will 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 will 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

The evaluator will test the encrypt functionality using a set of 100 plaintext, IV, and key 3-tuples. The keys, plaintext, and IV values are each 256-bits. For each 3-tuple, 1000 iterations will 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]
```

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

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

Evaluator Assessment:

Testing is satisfied through the Cryptographic Algorithm Verification Program (CAVP). The cryptographic implementation used by the TOE received CAVP Certificate A6141.

The following content should be included if:

- AES-CTR is selected from FCS_COP.1.1/ENCRYPT

AES-CTR Tests

Known Answer Test (KATs)

There are four Known Answer Tests (KATs) described below. For all KATs, the plaintext, initialization vector (IV), and ciphertext values shall be 256-bit blocks. The results from each test may either be obtained by the validator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator will compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

- Test 9: To test the encrypt functionality, the evaluator will supply 5 plaintext values and obtain the ciphertext value that results from encryption of the given plaintext using a 256-bit key value of all zeros and an IV of all zeros. To test the decrypt functionality, the evaluator will perform the same test as for encrypt, using the 5 ciphertext values as input.
- Test 10: To test the encrypt functionality, the evaluator will supply 5 256-bit key values and obtain the ciphertext value that results from encryption of an all zeros plaintext using the given key value and an IV of all zeros. To test the decrypt functionality, the evaluator will perform the same test as for encrypt, using an all zero ciphertext value as input.
- Test 11: To test the encrypt functionality, the evaluator will supply a set of key values described below and obtain the ciphertext values that result from AES encryption of an all zeros plaintext using the given key values and an IV of all zeros. The set of keys shall have shall have 256 256-bit keys. Keyi shall have the leftmost i bits be ones and the rightmost 256-i bits be zeros, for i in [1, N]. To test the decrypt functionality, the evaluator will supply the set of key and ciphertext value pairs described below and obtain the plaintext value that results from decryption of the given ciphertext using the given key values and an IV of all zeros. The set of key/ciphertext pairs shall have 256 256-bit pairs. Keyi shall have the leftmost i bits be ones and the rightmost 256-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.
- Test 12: To test the encrypt functionality, the evaluator will supply the set of 256 plaintext values described below and obtain the two ciphertext values that result from encryption of the given plaintext using a 256 bit key value of all zeros, respectively, and an IV of all zeros. Plaintext value i in each set shall have the leftmost bits be ones and the rightmost 256-i bits be zeros, for i in [1, 256]. To test the decrypt functionality, the evaluator will perform the same test as for encrypt, using ciphertext values of the same form as the plaintext in the encrypt test as input.

Multi-Block Message Test

The evaluator will test the encrypt functionality by encrypting an i-block message where 1 less-than i less-than-or-equal to 10. For each i the evaluator will choose a key, IV, and plaintext message of length i blocks and encrypt the message, using the mode to be tested, with the chosen key. 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. The evaluator will also test the decrypt functionality by decrypting an i-block message where 1 less-than i less-than-or-equal to 10. For each i the evaluator will choose a key and a ciphertext message of length i blocks and decrypt the message, using the mode to be tested, with the chosen key. The plaintext shall be compared to the result of decrypting the same ciphertext message with the same key using a known good implementation.

Monte-Carlo Test

For AES-CTR mode perform the Monte Carlo Test for ECB Mode on the encryption engine of the counter mode implementation. There is no need to test the decryption engine.

The evaluator will test the encrypt functionality using 100 plaintext/key pairs. Each key shall be 256-bit. The plaintext

values shall be 256-bit blocks. For each pair, 1000 iterations shall be run as follows:

For AES-ECB mode

Input: PT, Key

for i = 1 to 1000:
CT[i] = AES-ECB-Encrypt(Key, PT)
PT = CT[i]

The ciphertext computed in the 1000th iteration 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.

Evaluator Assessment:

Testing is satisfied through the Cryptographic Algorithm Verification Program (CAVP). The cryptographic implementation used by the TOE received CAVP Certificate A6141.

The following content should be included if:

- AES Key Wrap (KW) (as defined in NIST SP 800-38F), AES Key Wrap with Padding (KWP) (as defined in NIST SP 800-38F) is selected from FCS_COP.1.1/ENCRYPT

AES Key Wrap (AES-KW) and Key Wrap with Padding (AES-KWP) Test

The evaluator will test the authenticated encryption functionality of AES-KW for EACH combination of the following input parameter lengths:

- 256 bit key encryption keys (KEKs)
- Three plaintext lengths. One of the plaintext lengths will be two semi-blocks (256 bits). One of the plaintext lengths will be three semi-blocks (192 bits). The third data unit length will be the longest supported plaintext length less than or equal to 64 semi-blocks (4096 bits).

using a set of 100 key and plaintext pairs and obtain the ciphertext that results from AES-KW authenticated encryption. To determine correctness, the evaluator will use the AES-KW authenticated-encryption function of a known good implementation.

The evaluator will test the authenticated-decryption functionality of AES-KW using the same test as for authenticated-encryption, replacing plaintext values with ciphertext values and AES-KW authenticated-encryption with AES-KW authenticated-decryption.

Evaluator Assessment:

N/A. The TOE does not claim AES-KW or AES-KWP

The following content should be included if:

- AES Key Wrap with Padding (KWP) (as defined in NIST SP 800-38F) is selected from FCS_COP.1.1/ENCRYPT

The evaluator will test the authenticated-encryption functionality of AES-KWP using the same test as for AES-KW authenticated-encryption with the following change in the three plaintext lengths:

- One plaintext length will be one octet. One plaintext length will be 20 octets (160 bits).
- One plaintext length will be the longest supported plaintext length less than or equal to 512 octets (4096 bits).

The evaluator will test the authenticated-decryption functionality of AES-KWP using the same test as for AES-KWP authenticated-encryption, replacing plaintext values with ciphertext values and AES-KWP authenticated-encryption with AES-KWP authenticated-decryption.

Evaluator Assessment:

N/A. The TOE does not claim AES-KWP

The following content should be included if:

- AES-CCM or AES-CCMP-256 is selected from FCS_COP.1.1/ENCRYPT

AES-CCM Tests

The evaluator will test the generation-encryption and decryption-verification functionality of AES-CCM for the following input parameter and tag lengths:

- 128 bit (if selected) and 256 bit keys.
- Two payload lengths. One payload length will be the shortest supported payload length, greater than or equal to zero bytes. The other payload length will be the longest supported payload length, less than or equal to 32 bytes (256 bits).
- Two or three associated data lengths. One associated data length will be 0, if supported. One associated data length will be the shortest supported payload length, greater than or equal to zero bytes. One associated data length will be the longest supported payload length, less than or equal to 32 bytes (256 bits). If the implementation supports an associated data length of 2 16 bytes, an associated data length of 216 bytes will be tested.
- Nonce lengths. The evaluator will test all nonce lengths between 7 and 13 bytes, inclusive, that are supported by the OS.
- Tag lengths. The evaluator will test all of the following tag length values that are supported by the OS: 4, 6, 8, 10, 12, 14 and 16 bytes.

To test the generation-encryption functionality of AES-CCM, the evaluator will perform the following four tests:

- Test 13: For EACH supported key and associated data length and ANY supported payload, nonce and tag

length, the evaluator will supply one key value, one nonce value and 10 pairs of associated data and payload values and obtain the resulting ciphertext.

- Test 14: For EACH supported key and payload length and ANY supported associated data, nonce and tag length, the evaluator will supply one key value, one nonce value and 10 pairs of associated data and payload values and obtain the resulting ciphertext.
- Test 15: For EACH supported key and nonce length and ANY supported associated data, payload and tag length, the evaluator will supply one key value and 10 associated data, payload and nonce value 3-tuples and obtain the resulting ciphertext.
- Test 16: For EACH supported key and tag length and ANY supported associated data, payload and nonce length, the evaluator will supply one key value, one nonce value and 10 pairs of associated data and payload values and obtain the resulting ciphertext.

To determine correctness in each of the above tests, the evaluator will compare the ciphertext with the result of generation-encryption of the same inputs with a known good implementation.

To test the decryption-verification functionality of AES-CCM, for EACH combination of supported associated data length, payload length, nonce length and tag length, the evaluator will supply a key value and 15 nonce, associated data and ciphertext 3-tuples and obtain either a FAIL result or a PASS result with the decrypted payload. The evaluator will supply 10 tuples that should FAIL and 5 that should PASS per set of 15.

Additionally, the evaluator will use tests from the IEEE 802.11-02/362r6 document "Proposed Test vectors for IEEE 802.11 TGi", dated September 10, 2002, Section 2.1 AESCCMP Encapsulation Example and Section 2.2 Additional AES CCMP Test Vectors to further verify the IEEE 802.11-2007 implementation of AES-CCMP.

Evaluator Assessment:

N/A. The TOE does not claim AES-CCM or AES-CCMP-256.

The following content should be included if:

- AES-GCMP-256 is selected from FCS_COP.1.1/ENCRYPT

AES-GCMP Test

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

- 256 bit keys
- Two plaintext lengths. One of the plaintext lengths will be a non-zero integer multiple of 256 bits, if supported. The other plaintext length will not be an integer multiple of 256 bits, if supported.
- Three AAD lengths. One AAD length will be 0, if supported. One AAD length will be a non-zero integer multiple of 256 bits, if supported. One AAD length will not be an integer multiple of 256 bits, if supported.
- Two IV lengths. If 96 bit IV is supported, 96 bits will be one of the two IV lengths tested.

The evaluator will 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 will 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.

The evaluator will 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 will include five tuples that Pass and five that Fail.

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 will compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

AES-GCMP Test Monte Carlo Tests

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

- 256 bit keys
- Two plaintext lengths. One of the plaintext lengths will be a non-zero integer multiple of 256 bits, if supported. The other plaintext length will not be an integer multiple of 256 bits, if supported.
- Three AAD lengths. One AAD length will be 0, if supported. One AAD length will be a non-zero integer multiple of 256 bits, if supported. One AAD length will not be an integer multiple of 256 bits, if supported.
- Two IV lengths. If 96 bit IV is supported, 96 bits will be one of the two IV lengths tested.

The evaluator will 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 will 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.

The evaluator will 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 will include five tuples that Pass and five that Fail.

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 will compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

Evaluator Assessment:

Testing is satisfied through the Cryptographic Algorithm Verification Program (CAVP). The cryptographic implementation used by the TOE received CAVP certificate A6141.

2.1.5 FCS_COP.1/HASH Cryptographic Operation – Hashing (Refined)

2.1.5.1 TSS Assurance Activity

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

Evaluator Assessment:

Table 19 in Section 7 of the [ST] states that the TOE supports cryptographic hashing services conforming to FIPS Pub 180-4. The hashing algorithms are associated with the following cryptographic functions:

- Signature services (FCS_COP.1/SIGN)
- HMAC services (FCS_COP.1/KEYHMAC
- SSH (FCS_SSH_EXT.1, FCS_SSHC_EXT.1)
- TLS (FCS_TLSC_EXT.1)
- Boot integrity (FPT_TST_EXT.1)
- Trusted updates (FPT_TUD_EXT.1, FPT_TUD_EXT.2)

The following hashing algorithms supported: SHA-256, SHA-384 and SHA-512.
The message digest sizes supported are: 256 bits, 384 bits and 512 bits.

2.1.5.2 Test Assurance Activity

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

The following tests require the developer to provide access to a test application that provides the evaluator with tools that are typically not found in the production application.

- Test 17: Short Messages Test (Bit oriented Mode) - The evaluator will generate 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 will be pseudorandomly generated. The evaluator will 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.
- Test 18: Short Messages Test (Byte oriented Mode) - The evaluator will generate 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 will be pseudorandomly generated. The evaluator will 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.
- Test 19: Selected Long Messages Test (Bit oriented Mode) - The evaluator will generate an input set consisting of m messages, where m is the block length of the hash algorithm. The length of the ith message is $512 + 99 \cdot i$, where $1 \leq i \leq m$. The message text will be pseudorandomly generated. The evaluator will 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.
- Test 20: Selected Long Messages Test (Byte oriented Mode) - The evaluator will generate an input set consisting of m/8 messages, where m is the block length of the hash algorithm. The length of the ith message is $512 + 8 \cdot 99 \cdot i$, where $1 \leq i \leq m/8$. The message text will be pseudorandomly generated. The evaluator will 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.
- Test 21: Pseudorandomly Generated Messages Test - This test is for byte-oriented implementations only. The evaluator will 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 evaluator will then formulate a set of 100 messages and associated digests by following the algorithm provided in Figure 1 of [SHAVS]. The evaluator will then ensure

that the correct result is produced when the messages are provided to the TSF.

Evaluator Assessment:

Testing is satisfied through the Cryptographic Algorithm Verification Program (CAVP). The cryptographic implementation used by the TOE received CAVP certificate A6141.

2.1.6 FCS_COP.1/SIGN Cryptographic Operation – Signing (Refined)

2.1.6.1 TSS Assurance Activity [TD0873]

[Conditional: if "2048-bit (for secure boot only) or greater" is selected] The evaluator shall check that the TSS documents that 2048-bit RSA is used only for secure boot and a greater key size is used for any other functions

Evaluator Assessment:

N/A. 2048-bit for secure boot only is not selected.

2.1.6.2 Test Assurance Activity [TD0873]

The following tests require the developer to provide access to a test application that provides the evaluator with tools that are typically not found in the production application.

The following content should be included if:

- ECDSA schemes is selected from FCS_COP.1.1/SIGN

ECDSA Algorithm Tests

- Test 22: ECDSA SP-800-186 Section 3 Signature Generation Test. For each supported NIST curve (i.e., P-384 and P-521) and SHA function pair, the evaluator will 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 will use the signature verification function of a known good implementation.
- Test 23: ECDSA SP-800-186 Section 3 Signature Verification Test. For each supported NIST curve (i.e., P-384 and P-521) and SHA function pair, the evaluator will 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 will verify that 5 responses indicate success and 5 responses indicate failure.

Evaluator Assessment:

Testing is satisfied through the Cryptographic Algorithm Verification Program (CAVP). The cryptographic implementation used by the TOE received CAVP certificate A6141.

The following content should be included if:

- RSA schemes using cryptographic key sizes of 2048-bit or greater that meet the following: FIPS PUB 186-5, "Digital Signature Standard (DSS)", Section 4 is selected from FCS_COP.1.1/SIGN

[Conditional: if “2048-bit (for secure boot only) or greater” is selected] The evaluator shall check that the TSS documents that 2048-bit RSA is used only for secure boot and a greater key size is used for any other functions.

RSA Signature Algorithm Tests

- Test 24: Signature Generation Test. The evaluator will verify the implementation of RSA Signature Generation by the OS 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 will have the OS use its private key and modulus value to sign these messages. The evaluator will verify the correctness of the TSF' signature using a known good implementation and the associated public keys to verify the signatures.
- Test 25: Signature Verification Test. The evaluator will perform the Signature Verification test to verify the ability of the OS to recognize another party's valid and invalid signatures. The evaluator will 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 evaluator will verify that the OS returns failure when validating each signature.

Evaluator Assessment:

Testing is satisfied through the Cryptographic Algorithm Verification Program (CAVP). The cryptographic implementation used by the TOE received CAVP certificate A6141.

2.1.6.3 Guidance Documentation Assurance Activity [TD0873]

[Conditional: if “2048-bit (for secure boot only) or greater” is selected] The evaluator shall check that the AGD documents any configuration needed to ensure 2048-bit RSA is used only for secure boot and a greater key size is used for any other functions.

Evaluator Assessment:

N/A. 2048-bit for secure boot only is not selected.

2.1.7 FCS_COP.1/KEYHMAC Cryptographic Operation – Keyed-Hash Message Authentication (Refined)

2.1.7.1 Test Assurance Activity

The evaluator will perform the following activities based on the selections in the ST.

For each of the supported parameter sets, the evaluator will compose 15 sets of test data. Each set consists of a key and message data. The evaluator will have the OS generate HMAC tags for these sets of test data. The resulting MAC

tags will be compared against the result of generating HMAC tags with the same key using a known-good implementation.

Evaluator Assessment:

Testing is satisfied through the Cryptographic Algorithm Verification Program (CAVP). The cryptographic implementation used by the TOE received CAVP certificate A6141.

2.1.8 FCS_RBG_EXT.1 Random Bit Generation

2.1.8.1 Test Assurance Activity

The evaluator will perform the following tests:

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

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 will 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 SP 800-90A).

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 will 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.

The following list contains more information on some of the input values to be generated/selected by the evaluator.

- **Entropy inp**ut: 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 less than or equal to 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 will 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.

Documentation will be produced - and the evaluator will perform the activities - in accordance with Appendix E - Entropy Documentation and Assessment and the Clarification to the Entropy Documentation and Assessment Annex.

In the future, specific statistical testing (in line with NIST SP 800-90B) will be required to verify the entropy estimates.

Evaluator Assessment:

Testing is satisfied through the Cryptographic Algorithm Verification Program (CAVP). The cryptographic implementation used by the TOE received CAVP certificate A6141.

2.1.9 FCS_STO_EXT.1 Storage of Sensitive Data

2.1.9.1 TSS Assurance Activity

The evaluator will check the TSS to ensure that it lists all persistent sensitive data for which the OS provides a storage capability. For each of these items, the evaluator will confirm that the TSS lists for what purpose it can be used, and how it is stored. The evaluator will confirm that cryptographic operations used to protect the data occur as specified in FCS_COP.1/ENCRYPT.

Evaluator Assessment:

Table 19 in Section 7 - TOE Summary Specification of [ST] states that the TOE includes Oracle Linux 9.3 OpenSSL which provides an interface to end users to securely store sensitive data on the filesystem. OpenSSL provides file encryption services using AES-CBC, AES-CTR, and AES-GCMP with 256-bit keys.

Sensitive data stored in the /etc directory are keys, user passwords and application credentials, along with system-wide configuration files and system databases. Strict file permissions and/or encryption ensure access only by the 'root' user and/or the application storing the sensitive data.

2.1.9.2 Guidance Documentation Assurance Activity

The evaluator will consult the developer documentation to verify that instructions exists on applications should securely store credentials.

Evaluator Assessment:

Section 15 in the [AGD] provides information on the secure storage of application credentials.

2.2 User Data Protection (FDP)

2.2.1 FDP_ACF_EXT.1 Access Controls for Protecting User Data

2.2.1.1 TSS Assurance Activity

The evaluator will confirm that the TSS comprehensively describes the access control policy enforced by the OS. The description must include the rules by which accesses to particular files and directories are determined for particular users. The evaluator will inspect the TSS to ensure that it describes the access control rules in such detail that given any possible scenario between a user and a file governed by the OS the access control decision is unambiguous.

Evaluator Assessment:

Table 19 in Section 7 of the [ST] identifies the access control policy enforced by the OS. The TOE provides support for POSIX type ACLs. Users can configure ACLs that define access for users, groups, programs, processes, files, and directories.

2.2.1.2 Test Assurance Activity

The evaluator will create two new standard user accounts on the system and conduct the following tests:

Test 26: The evaluator will authenticate to the system as the first user and create a file within that user's home directory. The evaluator will then log off the system and log in as the second user. The evaluator will then attempt to read the file created in the first user's home directory. The evaluator will ensure that the read attempt is denied.

Test 27: The evaluator will authenticate to the system as the first user and create a file within that user's home directory. The evaluator will then log off the system and log in as the second user. The evaluator will then attempt to modify the file created in the first user's home directory. The evaluator will ensure that the modification is denied.

Test 28: The evaluator will authenticate to the system as the first user and create a file within that user's user directory. The evaluator will then log off the system and log in as the second user. The evaluator will then attempt to delete the file created in the first user's home directory. The evaluator will ensure that the deletion is denied.

Test 29: The evaluator will authenticate to the system as the first user. The evaluator will attempt to create a file in the second user's home directory. The evaluator will ensure that the creation of the file is denied.

Test 30: The evaluator will authenticate to the system as the first user and attempt to modify the file created in the first user's home directory. The evaluator will ensure that the modification of the file is accepted.

Test 31: The evaluator will authenticate to the system as the first user and attempt to delete the file created in the first user's directory. The evaluator will ensure that the deletion of the file is accepted.

Evaluator Assessment:

Test 26: The evaluator created a file while logged in as the first user. After re-authenticating as a different user, that user was unable to read the file created by the first user.

Test 27: The evaluator created a file while logged in as the first user. After re-authenticating as a different user, that user was unable to modify the file created by the first user.

Test 28: The evaluator created a file while logged in as the first user. After re-authenticating as a different user, that user was unable to delete the file created by the first user.

Test 29: The evaluator authenticated to the system as the first user and attempted to create a file in the home directory of a different user. The evaluator verified that this action was denied.

Test 30: The evaluator authenticated to the system as the first user and attempted to modify a file in that user's home directory. The evaluator verified the modification of the file was accepted.

Test 31: The evaluator authenticated to the system as the first user and attempted to delete a file in that user's home directory. The evaluator verified the deletion of the file was accepted.

2.3 Security Management (FMT)

2.3.1 FMT_MOF_EXT.1 Management of security functions behavior

2.3.1.1 TSS Assurance Activity

The evaluator will verify that the TSS describes those management functions that are restricted to Administrators, including how the user is prevented from performing those functions, or not able to use any interfaces that allow access to that function.

Evaluator Assessment:

Table 19 in Section 7 of the [ST] identifies the restricted management functions (FMT_SMF_EXT.1 section) and how those functions are restricted by the operating system (FMT_MOF_EXT.1 section).

The following management functions are restricted to the administrator:

- Enable/disable session timeout
- Configure session inactivity timeout
- Configure local audit storage capacity
- Configure minimum password length
- Configure minimum number of special characters in password
- Configure minimum number of numeric characters in password
- Configure minimum number of uppercase characters in password
- Configure minimum number of lowercase characters in password
- Configure lockout policy for unsuccessful authentication attempts through limiting number of attempts during a time period
- Configure host-based firewall
- Configure name/address of audit/logging server to which to send audit/logging records
- Configure audit rules
- Configure name/address of network time server
- Enable/disable automatic software update

2.3.1.2 Test Assurance Activity

The evaluator will also perform the following test.

- Test 32: For each function that is indicated as restricted to the administrator, the evaluation will perform the function as an administrator, as specified in the Operational Guidance, and determine that it has the expected effect as outlined by the Operational Guidance and the SFR. The evaluator will then perform the function (or otherwise attempt to access the function) as a non-administrator and observe that they are unable to invoke that functionality.

Evaluator Assessment:

Test 32: The evaluator attempted to perform each of the functions both as an administrator and a non-administrator as specified in the Operational Guidance and the SFR. The evaluator verified that configuration of these functions is only available to administrators.

2.3.2 FMT_SMF_EXT.1 Specification of Management Functions

2.3.2.1 Guidance Documentation Assurance Activity

The evaluator will verify that every management function captured in the ST is described in the operational guidance and that the description contains the information required to perform the management duties associated with the management function.

Evaluator Assessment:

The evaluator verified that the [AGD] document provides the information required to perform the management duties associated with management functions in the following sections:

Section 8 Configuring User Authentication

Section 10 Creating User Accounts

Section 12 System Firewall

Section 13 Network Time Service

Section 14 Session Timeout

Section 18 Applying Updates

Section 19 Auditing

2.3.2.2 Test Assurance Activity

The evaluator will test the OS's ability to provide the management functions by configuring the operating system and testing each option selected from above. The evaluator is expected to test these functions in all the ways in which the ST and guidance documentation state the configuration can be managed.

Evaluator Assessment:

This test is satisfied by FMT_MOF_EXT.1

2.4 Protection of the TSF (FPT)

2.4.1 FPT_ACF_EXT.1 Access Controls

2.4.1.1 TSS Assurance Activity

The evaluator will confirm that the TSS specifies the locations of kernel drivers/modules, security audit logs, shared libraries, system executables, and system configuration files. Every file does not need to be individually identified, but the system's conventions for storing and protecting such files must be specified.

Evaluator Assessment:

The TSS lists the following locations in Table 19 in Section 7 of the [ST] for each identified file type from the assurance activity:

- /boot, /usr/lib/firmware, usr/lib/modules: contain kernel, kernel modules and device drivers
- /var/log/audit, /var/log/secure: contain audit data
- /usr/lib and /usr/lib64: contain shared libraries
- /usr/bin, /usr/sbin, /usr/libexec: contain system executables
- /etc, /usr/lib: contain system configuration files.

The OS implements access control for these folders which prohibits unprivileged users from reading security audit logs and system-wide credential repositories.

2.4.1.2 Test Assurance Activity

The evaluator will create an unprivileged user account. Using this account, the evaluator will ensure that the following tests result in a negative outcome (i.e., the action results in the OS denying the evaluator permission to complete the action):

- Test 33: The evaluator will attempt to modify all kernel drivers and modules.
- Test 34: The evaluator will attempt to modify all security audit logs generated by the logging subsystem.
- Test 35: The evaluator will attempt to modify all shared libraries that are used throughout the system.
- Test 36: The evaluator will attempt to modify all system executables.
- Test 37: The evaluator will attempt to modify all system configuration files.
- Test 38: The evaluator will attempt to modify any additional components selected.

The evaluator will create an unprivileged user account. Using this account, the evaluator will ensure that the following tests result in a negative outcome (i.e., the action results in the OS denying the evaluator permission to complete the action):

- Test 39: The evaluator will attempt to read security audit logs generated by the auditing subsystem
- Test 40: The evaluator will attempt to read system-wide credential repositories
- Test 41: The evaluator will attempt to read any other object specified in the assignment

Evaluator Assessment:

Test 33: The evaluator attempted to modify all kernel drivers and modules using an unprivileged user account and was denied permission.

Test 34: The evaluator attempted to modify all security audit logs generated by the logging subsystem using an unprivileged user account and was denied permission.

Test 35: The evaluator attempted to modify all shared libraries that are used throughout the system using an unprivileged user account and was denied permission.

Test 36: The evaluator attempted to modify all system executables using an unprivileged user account and was denied permission.

Test 37: The evaluator attempted to modify all system configuration files using an unprivileged user account and was denied permission.

Test 38: N/A. Additional components are not selected.

Test 39: The evaluator attempted to read security audit logs using an unprivileged user account and was denied permission.

Test 40: The evaluator attempted to read system-wide credential repositories using an unprivileged user account and was denied permission.

Test 41: N/A. No other objects are specified.

2.4.2 FPT_ASLR_EXT.1 Address Space Layout Randomization

2.4.2.1 Test Assurance Activity

The evaluator will select 3 executables included with the TSF. If the TSF includes a web browser it must be selected. If the TSF includes a mail client it must be selected. For each of these apps, the evaluator will launch the same executables on two separate instances of the OS on identical hardware and compare all memory mapping locations. The evaluator will ensure that no memory mappings are placed in the same location. If the rare chance occurs that two mappings are the same for a single executable and not the same for the other two, the evaluator will repeat the test with that executable to verify that in the second test the mappings are different. This test can also be completed on the same hardware and rebooting between application launches.

Evaluator Assessment:

The evaluator chose 3 executables included with the TSF and determined the memory mapping of each. After re-booting the TOE, the evaluator verified that the memory mappings were different for each executable.

2.4.3 FPT_SBOP_EXT.1 Stack Buffer Overflow Protection

2.4.3.1 TSS Assurance Activity

For stack-based OSes, the evaluator will determine that the TSS contains a description of stack-based buffer overflow protections used by the OS. These are referred to by a variety of terms, such as stack cookie, stack guard, and stack canaries. The TSS must include a rationale for any binaries that are not protected in this manner.

For OSes that store parameters/variables separately from control flow values, the evaluator will verify that the TSS describes what data structures control values, parameters, and variables are stored. The evaluator will also ensure that the TSS includes a description of the safeguards that ensure parameters and variables do not intermix with control flow values.

Evaluator Assessment:

The evaluator verified that the TSS contains a description of the stack-based buffer overflow protections used by the OS. The OS implements compiler flag stack-based buffer overflow protections (fstack-protector-strong). The FPT_SBOP_EXT.1 entry in Table 19 in Section 7 of the [ST] provides a list of libraries that were not compiled with stack-based protections and rationale as to why the protections are not required.

2.4.3.2 Test Assurance Activity

For stack-based OSes, the evaluator will perform the following test:

Test 42: The evaluator will inventory the kernel, libraries, and application binaries to determine those that do not implement stack-based buffer overflow protections. This list should match up with the list provided in the TSS.

Evaluator Assessment:

The evaluator analyzed the kernel and application libraries to determine which ones did not include stack-based buffer overflow protections. The list provided in the TSS in Section 7 of the [ST] matches the results found by the evaluator.

2.4.4 FPT_TST_EXT.1 Boot Integrity

2.4.4.1 TSS Assurance Activity

The evaluator will verify that the TSS section of the ST includes a comprehensive description of the boot procedures, including a description of the entire bootchain, for the TSF. The evaluator will ensure that the OS cryptographically verifies each piece of software it loads in the bootchain to include bootloaders and the kernel. Software loaded for execution directly by the platform (e.g. first-stage bootloaders) is out of scope. For each additional category of executable code verified before execution, the evaluator will verify that the description in the TSS describes how that software is cryptographically verified.

The evaluator will verify that the TSS contains a description of the protection afforded to the mechanism performing the cryptographic verification.

Evaluator Assessment:

The evaluator verified that the TSS in Table 19 of Section 7 of the [ST] includes a detailed description of the procedures that occur on boot for the operating system. On boot, the BIOS performs self-tests and then reads the Master Boot Record. Two stages of bootloader then follow (the first stage verifies the keys for GRUB2, and then GRUB2 is loaded). The bootloader then loads the kernel into memory. The kernel is then responsible for loading all the driver modules and executing initial processes (e.g., systemd, /sbin/init). The user space starts after the kernel boot process is complete. The software is tested for integrity using HMAC-SHA-256.

For more detail on the boot process, please see Table 19 (FPT_TST_EXT.1) in Section 7 of the [ST].

2.4.4.2 Test Assurance Activity

The evaluator will also perform the following tests:

- Test 43: The evaluator will perform actions to cause TSF software to load and observe that the integrity mechanism does not flag any executables as containing integrity errors and that the OS properly boots.
- Test 44: The evaluator will modify a TSF executable that is part of the bootchain verified by the TSF (i.e. Not the first-stage bootloader) and attempt to boot. The evaluator will ensure that an integrity violation is triggered and the OS does not boot (Care must be taken so that the integrity violation is determined to be the cause of the failure to load the module, and not the fact that in such a way to invalidate the structure of the module.).
- Test 45[conditional, to be performed if
    - a digital signature using an X509 certificate with hardware-based protection is selected from FPT_TST_EXT.1.1

  ]: If the ST author indicates that the integrity verification is performed using public key in an X509 certificate, the evaluator will verify that the boot integrity mechanism includes a certificate validation according to FIA_X509_EXT.1 for all certificates in the chain from the certificate used for boot integrity to a certificate in the trust store that are not themselves in the trust store. This means that, for each X509 certificate in this chain that is not a trust store element, the evaluator must ensure that revocation information is available to the TOE during the bootstrap mechanism (before the TOE becomes fully operational).

Evaluator Assessment:

Test 43: The evaluator booted the device and verified that the TOE loads properly without any errors.

Test 44: The evaluator modified a file that was part of the bootchain and verified that an integrity violation occurs on boot of the TOE. The OS did not boot after the modification was performed.

Test 45: N/A. The TOE uses a digital signature to verify integrity of the boot process. X509 certificates are not used for this process.

2.4.5 FPT_TUD_EXT.1 Trusted Update

2.4.5.1 Test Assurance Activity

The evaluator will check for an update using procedures described in the documentation and verify that the OS provides a list of available updates. Testing this capability may require installing and temporarily placing the system into a configuration in conflict with secure configuration guidance which specifies automatic update.

The evaluator is also to ensure that the response to this query is authentic by using a digital signature scheme specified in FCS_COP.1/SIGN. The digital signature verification may be performed as part of a network protocol occurs over a trusted channel as described in FTP_ITC_EXT.1.) If the signature verification is not performed as part of a trusted channel, the evaluator will send a query response with a bad signature and verify that the signature verification fails. The evaluator will then send a query response with a good signature and verify that the signature verification is successful.

For the following tests, the evaluator will initiate the download of an update and capture the update prior to installation. The download could originate from the vendor's website, an enterprise-hosted update repository, or another system (e.g. network peer). All supported origins for the update must be indicated in the TSS and evaluated.

- Test 46: The evaluator will ensure that the update has a digital signature belonging to the vendor prior to its installation. The evaluator will modify the downloaded update in such a way that the digital signature is no longer valid. The evaluator will then attempt to install the modified update. The evaluator will ensure that the OS does not install the modified update.
- Test 47: The evaluator will ensure that the update has a digital signature belonging to the vendor. The evaluator will then attempt to install the update (or permit installation to continue). The evaluator will ensure that the OS successfully installs the update.

Evaluator Assessment:

Test 46: The evaluator confirmed that the package has a digital signature belonging to the vendor. The evaluator then modified the update and attempted to install the modified update. The OS successfully rejected the modified update.

Test 47: The evaluator confirmed that the package has a digital signature belonging to the vendor. The evaluator then attempted to install the update. The OS successfully installed the update.

2.4.6 FPT_TUD_EXT.2 Trusted Update for Application Software

2.4.6.1 Test Assurance Activity

The evaluator will check for updates to application software using procedures described in the documentation and verify that the OS provides a list of available updates. Testing this capability may require temporarily placing the system into a configuration in conflict with secure configuration guidance which specifies automatic update.

The evaluator is also to ensure that the response to this query is authentic by using a digital signature scheme specified in FCS_COP.1/SIGN. The digital signature verification may be performed as part of a network protocol as described in FTP_ITC_EXT.1. If the signature verification is not performed as part of a trusted channel, the evaluator will send a query response with a bad signature and verify that the signature verification fails. The evaluator will then

send a query response with a good signature and verify that the signature verification is successful.

The evaluator will initiate an update to an application. This may vary depending on the application, but it could be through the application vendor's website, a commercial app store, or another system. All origins supported by the OS must be indicated in the TSS and evaluated. However, this only includes those mechanisms for which the OS is providing a trusted installation and update functionality. It does not include user or administrator-driven download and installation of arbitrary files.

- Test 48: The evaluator will ensure that the update has a digital signature which chains to the OS vendor or another trusted root managed through the OS. The evaluator will modify the downloaded update in such a way that the digital signature is no longer valid. The evaluator will then attempt to install the modified update. The evaluator will ensure that the OS does not install the modified update.
- Test 49: The evaluator will ensure that the update has a digital signature belonging to the OS vendor or another trusted root managed through the OS. The evaluator will then attempt to install the update. The evaluator will ensure that the OS successfully installs the update.

Evaluator Assessment:

Test 48: The evaluator confirmed that the package has a digital signature belonging to the vendor. The evaluator then modified the update and attempted to install the modified update. The OS successfully rejected the modified update.

Test 49: The evaluator confirmed that the package has a digital signature belonging to the vendor. The evaluator then attempted to install the update. The OS successfully installed the update.

2.5 Audit Data Generation (FAU)

2.5.1 FAU_GEN.1 Audit Data Generation (Refined)

2.5.1.1 Guidance Assurance Activities

The evaluator will check the administrative guide and ensure that it lists all of the auditable events. The evaluator will check to make sure that every audit event type selected in the ST is included.

The evaluator will check the administrative guide and ensure that it provides a format for audit records. Each audit record format type must be covered, along with a brief description of each field. The evaluator will ensure that the fields contains the information required.

Evaluator Assessment:

Section 19 of the [AGD] lists all the auditable events. The evaluator verified that the audit event types listed in the ST are included in the [AGD].

Section 19 of the [AGD] provides samples of each type of audit record and a description of each of the fields for these records.

2.5.1.2 Test Assurance Activity [TD0693]

The evaluator will test the OS's ability to correctly generate audit records by having the TOE generate audit records for the events listed in the ST. This should include all instance types of an event specified. When verifying the test results, the evaluator will ensure the audit records generated during testing match the format specified in the administrative guide, and that the fields in each audit record provide the required information.

Evaluator Assessment:

The evaluator used various configuration changes and actions to cause the TOE to generate audit records. The evaluator verified that the audit records that were generated match the format that was specified in the guidance.

2.6 Identification and Authentication (FIA)

2.6.1 FIA_AFL.1 Authentication failure handling (Refined)

2.6.1.1 Test Assurance Activity [TD0691]

The evaluator will set an administrator-configurable threshold for failed attempts, or note the ST-specified assignment. The evaluator will then (per selection) repeatedly attempt to authenticate with an incorrect password, PIN, or certificate until the number of attempts reaches the threshold. Note that the authentication attempts and lockouts must also be logged as specified in FAU_GEN.1.

- Test 53 [conditional, to be performed if “authentication based on user name and password” is selected in FIA_AFL.1 and FIA_UAU.5]: The evaluator will attempt to authenticate repeatedly to the system with a known bad password. Once the defined number of failed authentication attempts has been reached the evaluator will ensure that the account that was being used for testing has had the actions detailed in the assignment list above applied to it. The evaluator will ensure that an event has been logged to the security event log detailing that the account has had these actions applied.
- Test 54 [conditional, to be performed if “authentication based on user name and a PIN that releases an asymmetric key stored in OE-protected storage is selected in FIA_AFL.1 and FIA_UAU.5]: The evaluator will attempt to authenticate repeatedly to the system with a known bad PIN. Once the defined number of failed authentication attempts has been reached the evaluator will ensure that the account that was being used for testing has had the actions detailed in the assignment list above applied to it. The evaluator will ensure that an event has been logged to the security event log detailing that the account has had these actions applied.
- Test 55 [conditional, to be performed if “authentication based on X.509 certificates” is selected in FIA_AFL.1 and FIA_UAU.5]: The evaluator will attempt to authenticate repeatedly to the system using a known bad certificate. Once the defined number of failed authentication attempts has been reached the evaluator will ensure that the account that was being used for testing has had the actions detailed in the assignment list above applied to it. The evaluator will ensure that an event has been logged to the security event log detailing that the account has had these actions applied.

Evaluator Assessment:

Test 53: The evaluator attempted to log into the TOE several times with a bad password and verified that after the configured threshold was passed, the user was locked out. The failed attempts and lockout were logged.

Test 54: N/A. A PIN is not utilized for authentication.

Test 55: N/A. Certificates are not utilized for authentication.

2.6.2 FIA_UAU.5 Multiple Authentication Mechanisms (Refined)

2.6.2.1 TSS Assurance Activity

The evaluator will ensure that the TSS describes the rules as to how each authentication mechanism specified in FIA_UAU.5.1 is implemented and used. Example rules are how the authentication mechanism authenticates the user (i.e. how does the TSF verify that the correct password or authentication factor is used), the result of a successful authentication (i.e. is the user input used to derive or unlock a key) and which authentication mechanism can be used at which authentication factor interfaces (i.e. if there are times, for example, after a reboot, that only specific authentication mechanisms can be used). Rules regarding how the authentication factors interact in terms of unsuccessful authentication are covered in FIA_AFL.1.

Evaluator Assessment:

Table 19 in Section 7 of the [ST] states that the TOE supports authentication based on username/password. The OS includes the Pluggable Authentication Module (PAM) authentication mechanism. PAM validates that a specified account is valid (and not expired), and that the password provided for that account is valid.

When a user attempts to login to the TOE, the login application calls the pam_unix module that is responsible for checking local account authentication. PAM will verify the user password against the values stored in /etc/shadow.

The TOE also utilizes public key based authentication which is described in the TSS.

2.6.2.2 Guidance Documentation Assurance Activity

The evaluator will verify that configuration guidance for each authentication mechanism is addressed in the AGD guidance.

Evaluator Assessment:

The evaluator examined the [AGD] document and verified that guidance for the configuration of authentication mechanisms is addressed in Section 8 - Configuring User Authentication and Section 5.3 Using Public Key Authentication.

2.6.2.3 Test Assurance Activity

The following content should be included if:

- authentication based on username and password is selected from FIA_UAU.5.1

- Test 56: The evaluator will attempt to authenticate to the OS using the known user name and password. The evaluator will ensure that the authentication attempt is successful.
- Test 57: The evaluator will attempt to authenticate to the OS using the known user name but an incorrect

password. The evaluator will ensure that the authentication attempt is unsuccessful.

The following content should be included if:

- username and a PIN that releases an asymmetric key is selected from FIA_UAU.5.1

The evaluator will examine the TSS for guidance on supported protected storage and will then configure the TOE or OE to establish a PIN which enables release of the asymmetric key from the protected storage (such as a TPM, a hardware token, or isolated execution environment) with which the OS can interface. The evaluator will then conduct the following tests:

- Test 58: The evaluator will attempt to authenticate to the OS using the known user name and PIN. The evaluator will ensure that the authentication attempt is successful.
- Test 59: The evaluator will attempt to authenticate to the OS using the known user name but an incorrect PIN. The evaluator will ensure that the authentication attempt is unsuccessful.

The following content should be included if:

- combination of authentication based on user name, password, and time-based one-time password is selected from FIA_UAU.5.1

The evaluator will configure the OS to authentication to authenticate to the OS using a username, password, and one-time password mechanism. The evaluator will then perform the following tests.

- Test 60: The evaluator will attempt to authenticate using a valid username, valid password, and valid one-time password. The evaluator will ensure that the authentication attempt is successful.
- Test 61: The evaluator will attempt to authenticate using a valid username, invalid password, and valid one-time password. The evaluator will ensure that the authentication attempt fails.
- Test 62: The evaluator will attempt to authenticate using a valid username, valid password, and invalid one-time password. The evaluator will ensure that the authentication attempt fails.
- Test 63: The evaluator will attempt to authenticate using a valid username, invalid password, and invalid one-time password. The evaluator will ensure that the authentication attempt fails.

Authentication mechanisms related to authentication based on X.509 certificates are tested under FIA_X509_EXT.1 and SSH public key-based authentication are tested in the Functional Package for Secure Shell (SSH), version 1.0.

For each authentication mechanism rule, the evaluator will ensure that the authentication mechanism(s) behave as documented in the TSS.

Evaluator Assessment:

Test 56: The evaluator attempted to authenticate to the OS using valid login credentials and was successful.

Test 57: The evaluator attempted to authenticate to the OS using a valid username with incorrect password and the attempt was unsuccessful.

Test 58: N/A. A PIN is not utilized for authentication.

Test 59: N/A. A PIN is not utilized for authentication.

Test 60: N/A. One time passwords are not utilized for authentication.

Test 61: N/A. One time passwords are not utilized for authentication.

Test 62: N/A. One time passwords are not utilized for authentication.

Test 63: N/A. One time passwords are not utilized for authentication.

2.6.3 FIA_X509_EXT.1 X.509 Certificate Validation

2.6.3.1 TSS Assurance Activity [TD0773]

The evaluator will 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.

If there are exceptional use cases where the OS cannot perform revocation checking in accordance with at least one of the revocation methods, the evaluator will ensure the TSS describes each revocation checking exception use case and, for each exception, the alternate functionality the TOE implements to determine the status of the certificate and disable functionality dependent on the validity of the certificate.

Evaluator Assessment:

Table 19 in Section 7 of the [ST] states that the certification validation process and certificate path validation algorithm used are defined in RFC 5280. The TOE validates x.509 certificates when they are presented as part of a TLS handshake.

2.6.3.2 Test Assurance Activity [TD0773]

The tests described must be performed in conjunction with the other certificate services evaluation activities, including the functions in FIA_X509_EXT.2.1. The evaluator will create a chain of at least four certificates: the node certificate to be tested, two Intermediate CAs, and the self-signed Root CA.

- Test 64: The evaluator will demonstrate that validating a certificate without a valid certification path results in the function failing, for each of the following reasons, in turn: by establishing a certificate path in which one of the issuing certificates is not a CA certificate, by omitting the basicConstraints field in one of the issuing certificates, by setting the basicConstraints field in an issuing certificate to have CA=False, by omitting the CA signing bit of the key usage field in an issuing certificate, and by setting the path length field of a valid CA field to a value strictly less than the certificate path. The evaluator will then establish a valid certificate path consisting of valid CA certificates, and demonstrate that the function succeeds. The evaluator will then remove trust in one of the CA certificates, and show that the function fails.
- Test 65: The evaluator will demonstrate that validating an expired certificate results in the function failing.
- Test 66: [Conditional, to be performed for use cases identified in exceptions that cannot be configured to allow revocation checking]. The evaluator will test that the OS can properly handle revoked certificates - conditional on whether CRL, OCSP, OCSP stapling, or OCSP multi-stapling is selected; if multiple methods are selected, then a test will be performed for each method. The evaluator will test revocation of the node certificate and revocation of the intermediate CA certificate (i.e. the intermediate CA certificate should be revoked by the root CA). If OCSP stapling per RFC 6066 is the only supported revocation method, testing

revocation of the intermediate CA certificate is omitted. The evaluator will 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. If the exceptions are configurable, the evaluator shall attempt to configure the exceptions to allow revocation checking for each function indicated in FIA_X509_EXT.2.

- Test 67: If any OCSP option is selected, the evaluator will 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 will 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.
- Test 68: The evaluator will 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.)
- Test 69: The evaluator will modify any byte in the last eight bytes of the certificate and demonstrate that the certificate fails to validate. (The signature on the certificate will not validate.)
- Test 70: The evaluator will modify any byte in the public key of the certificate and demonstrate that the certificate fails to validate. (The signature of the certificate will not validate.)
- Test 71[conditional, to be performed if
    - ECDSA schemes is selected from FCS_COP.1.1/SIGN
    - 6187 is selected from FCS_SSH_EXT.1.1 from Functional Package for Secure Shell (SSH), version 1.0

  ]:

    - Test 71.1: The evaluator will establish a valid, trusted certificate chain consisting of an EC leaf certificate, an EC Intermediate CA certificate not designated as a trust anchor, and an EC certificate designated as a trusted anchor, where the elliptic curve parameters are specified as a named curve. The evaluator will confirm that the TOE validates the certificate chain.
    - Test 71.2: The evaluator will replace the intermediate certificate in the certificate chain for Test 71.1 with a modified certificate, where the modified intermediate CA has a public key information field where the EC parameters uses an explicit format version of the Elliptic Curve parameters in the public key information field of the intermediate CA certificate from Test 71.1, and the modified Intermediate CA certificate is signed by the trusted EC root CA, but having no other changes. The evaluator will confirm the TOE treats the certificate as invalid.
- Test 72[conditional, to be performed if
    - exceptional use cases is selected from FIA_X509_EXT.1.1

  ]: For each exceptional use case for revocation checking described in the ST, the evaluator shall attempt to establish the conditions of the use case, designate the certificate as invalid and perform the function relying on the certificate. The evaluator shall observe that the alternate revocation checking mechanism successfully prevents performance of the function.

[Conditional, to be performed if “authentication based on X.509 certificates” is selected in FIA_UAU.5]:

The evaluator will generate an X.509v3 certificate for a user with the Client Authentication Extended Key Usage field set. The evaluator will provision the OS for authentication with the X.509v3 certificate. The evaluator will ensure that the certificates are validated by the OS as per FIA_X509_EXT.1.1 and then conduct the following two tests:

- Test 73: The evaluator will attempt to authenticate to the OS using the X.509v3 certificate. The evaluator will ensure that the authentication attempt is successful.
- Test 74: The evaluator will generate a second certificate identical to the first except for the public key and any values derived from the public key. The evaluator will attempt to authenticate to the OS with this

certificate. The evaluator will ensure that the authentication attempt is unsuccessful.

The tests described must be performed in conjunction with the other certificate services evaluation activities, including the functions in FIA_X509_EXT.2.1. The evaluator will create a chain of at least four certificates: the node certificate to be tested, two Intermediate CAs, and the self-signed Root CA.

- Test 75: The evaluator will construct a certificate path, such that the certificate of the CA issuing the OS's certificate does not contain the basicConstraints extension. The validation of the certificate path fails.
- Test 76: The evaluator will construct a certificate path, such that the certificate of the CA issuing the OS's certificate has the CA flag in the basicConstraints extension not set. The validation of the certificate path fails.
- Test 77: The evaluator will construct a certificate path, such that the certificate of the CA issuing the OS's certificate has the CA flag in the basicConstraints extension set to TRUE. The validation of the certificate path succeeds.

Evaluator Assessment:

Test 64:

The evaluator configured the TOE to connect to a remote TLS server that offers a CA certificate using a certification path in which one of the issuing certificates was not a CA certificate and verified the connection failed.

The evaluator configured the TOE to connect to a remote TLS server that offers a CA certificate using a certification path in which the basicConstraints field was omitted in one of the certificates and verified the connection failed.

The evaluator configured the TOE to connect to a remote TLS server that offers a CA certificate using a certification path in which the basicConstraints field included CA=false in one of the certificates and verified the connection failed.

The evaluator configured the TOE to connect to a remote TLS server that offers a CA certificate using a certification path in which the CA signing bit was omitted in one of the certificates and verified the connection failed.

The evaluator configured the TOE to connect to a remote TLS server that offers a CA certificate using a certification path in which the path length field in one of the certificates was set to a value less than the certification path. The connection failed.

The evaluator established a connection to a remote TLS server with a certificate path consisting of valid CA certificates and verified the connection succeeds.

The evaluator attempted to connect to a remote TLS server with the trust in one of the CA certificates removed and verified the connection fails.

Test 65:

The evaluator attempted to connect to a remote TLS server using an expired certificate and verified that the connection failed.

Test 66:

The evaluator confirmed that a successful connection can be made to a remote TLS server with a valid certificate chain. The evaluator then attempted to connect to a remote TLS server in which the intermediate CA certificate has been revoked and verified that the connection fails.

Testing of the revoked node certificate is satisfied by FCS_TLSC_EXT.1.3 Test #2

Test 67:

The evaluator configured the CA to sign a CRL with a certificate that did not have the cRLsign key usage bit set. Attempting to validate the CRL when connecting to the TLS server failed.

Test 68:

The evaluator attempted to make a connection to a remote TLS server that presents a certificate in which one of the first eight bytes is modified. The certificate fails to validate, and the connection fails.

Test 69:

The evaluator attempted to make a connection to a remote TLS server that presents a certificate in which one of the last eight bytes is modified. The certificate fails to validate, and the connection fails.

Test 70:

The evaluator attempted to make a connection to a remote TLS server that presents a certificate in which one of the bytes in the public key of the certificate is modified. The signature of the certificate does not validate, and the connection fails.

Test 71.1:

The evaluator attempted to make a connection to a remote TLS server that presents a certificate chain where the elliptic curve parameters are specified as a named curve. The connection is successful.

Test 71.2:

The evaluator attempted to make a connection to a remote TLS server that presents a certificate chain where a modified intermediate CA has a public key information field where the EC parameters uses an explicit format version of the EC parameters in the public key information field of the intermediate CA certificate used in Test 71.1. The connection fails.

Test 72: N/A. Exceptional use cases are not selected.

2.6.4 FIA_X509_EXT.2 X.509 Certificate Authentication

2.6.4.1 Test Assurance Activity [TD0789]

The evaluator will acquire or develop an application that uses the selected OS mechanism with an X.509v3 certificate. The evaluator will then run the application and ensure that the provided certificate is used to authenticate the connection.

The evaluator will repeat the activity for all selections listed.

Evaluator Assessment:

This test is satisfied by FCS_TLSC_EXT.1 Test #1.

2.7 Trusted Path/Channels (FTP)

2.7.1 FTP_ITC_EXT.1 Trusted channel communications

2.7.1.1 Test Assurance Activity [TD0789]

The evaluator shall configure the OS to communicate with another trusted IT product as identified in the third selection. The evaluator shall monitor network traffic while the OS performs communication with each of the servers identified in the third selection. The evaluator shall ensure that for each session a trusted channel was established in conformance with the selected protocols.

Evaluator Assessment:

This test is satisfied by FCS_TLSC_EXT.1 Test #1, FCS_SSHC_EXT.1.1 and FCS_SSHS_EXT.1.1.

2.7.2 FTP_TRP.1 Trusted Path

2.7.2.1 TSS Assurance Activity [TD0839]

The evaluator will examine the TSS to determine that the methods of remote or local OS administration are indicated, along with how those communications are protected. (Conditional: if “remote” is selected in FTP_TRP1.1) The evaluator will also confirm that all protocols listed in the TSS in support of OS administration are consistent with those specified in the requirement, and are included in the requirements in the ST.

Evaluator Assessment:

Table 19 in Section 7 of the [ST] states that the methods of remote administration supported are a command line interface available through SSHv2. The protocols listed in the TSS are consistent with those specified in the requirement

2.7.2.2 Guidance Documentation Assurance Activity [TD0839]

The evaluator will confirm that the operational guidance contains instructions for establishing the remote administrative sessions or initial user authentication for each supported method.

Evaluator Assessment:

Section 5 of the [AGD] provides instructions for establishing the remote administrative sessions.

2.7.2.3 Test Assurance Activity [TD0839]

The evaluator will also perform the following tests:

- Test 78: The evaluator will ensure that communications using each remote or local administration method is tested during the course of the evaluation, setting up the connections or initial user authentication as described in the operational guidance and ensuring that communication is successful.
- Test 79 (Conditional: if “remote” is selected in FTP_TRP1.1): For each method of remote administration supported, the evaluator will follow the operational guidance to ensure that there is no available interface that can be used by a remote user to establish a remote administrative sessions without invoking the trusted path.
- Test 80 (Conditional: if “remote” is selected in FTP_TRP1.1): The evaluator will ensure, for each method of remote administration, the channel data is not sent in plaintext.
- Test 81 (Conditional: if “remote” is selected in FTP_TRP1.1): The evaluator will ensure, for each method of remote administration, modification of the channel data is detected by the OS.

Evaluator Assessment:

Test 78: This test is satisfied by FCS_SSH_EXT.1.

Test 79: The evaluator has verified that the TOE only allows the trusted path to be used for remote administration over SSH.

Test 80: Testing for FCS_SSH_EXT.1 satisfies this requirement.

Test 81: The evaluator initiated an SSH connection and then used a Man-In-The-Middle attack to modify a packet. The modification of the packet was detected by the OS.

3. APPENDIX A - OPTIONAL REQUIREMENTS

A.1 Strictly Optional Requirements

None Selected.

A.2 Objective Requirements

None Selected.

4. APPENDIX B - SELECTION-BASED REQUIREMENTS

4.1 User Data Protection (FDP)

4.1.1 FDP_IFC_EXT.1 Information flow control

4.1.1.1 TSS Assurance Activity

The evaluator will verify that the TSS section of the ST describes the routing of IP traffic when a VPN client is enabled. The evaluator will ensure that the description indicates which traffic does not go through the VPN and which traffic does, and that a configuration exists for each in which only the traffic identified by the ST author as necessary for establishing the VPN connection (IKE traffic and perhaps HTTPS or DNS traffic) is not encapsulated by the VPN protocol (IPsec).

Evaluator Assessment:

Table 19 in Section 7 of the [ST] states that the TOE provides the XFRM framework with the XFRM netlink interface and it also provides the TUN/TAP interface for supporting user-space VPN clients operating at ISO/OSI level 2 or 3. Only IP traffic goes through the VPN and other traffic (DNS, etc) do not go through the VPN.

4.1.1.2 Test Assurance Activity

The evaluator will perform the following test:

- Test 94:
  - Step 1: The evaluator will enable a network connection. The evaluator will sniff packets while performing running applications that use the network such as web browsers and email clients. The evaluator will verify that the sniffer captures the traffic generated by these actions, turn off the sniffing tool, and save the session data.
  - Step 2: The evaluator will configure an IPsec VPN client that supports the routing specified in this requirement. The evaluator will turn on the sniffing tool, establish the VPN connection, and perform the same actions with the device as performed in the first step. The evaluator will verify that the sniffing tool captures traffic generated by these actions, turn off the sniffing tool, and save the session data.
  - Step 3: The evaluator will examine the traffic from both step one and step two to verify that all non-excepted Data Plane traffic in Step 2 is encapsulated by IPsec. The evaluator will examine the Security Parameter Index (SPI) value present in the encapsulated packets captured in Step 2 from the TOE to the Gateway and will verify this value is the same for all actions used to generate traffic through the VPN. Note that it is expected that the SPI value for packets from the Gateway to the TOE is different than the SPI value for packets from the TOE to the Gateway.
  - Step 4: The evaluator will perform a ping on the TOE host on the local network and verify that no packets sent are captured with the sniffer. The evaluator will attempt to send packets to the TOE outside the VPN tunnel (i.e. not through the VPN gateway), including from the local network, and verify that the TOE discards them.

Evaluator Assessment:

Test 94:

Step 1: The evaluator used a packet analyzer to capture traffic generated by an HTTP request and verified that the packets were captured.

Step 2: The evaluator repeated the actions performed in step 1 of the test after configuring the IPsec VPN client and verified that the generated traffic was captured.

Step 3: The evaluator examined the traffic generated in step 2 and verified that all non-excepted Data Plane traffic is encapsulated by IPsec. The SPI values present in the encapsulated packets were examined and verified to be accurate.

Step 4: The evaluator performed a ping on the TOE host from a host on the local network and verified that the there was no response from the TOE in the packet capture. The evaluator attempted to send packets to the TOE outside the VPN tunnel and verified that they are discarded.

5. Security Assurance Requirement Activities

5.1 ADV: Development

5.1.1 Basic Functional Specification (ADV_FSP.1)

5.1.1.1 Evaluation Activity:

There are no specific evaluation activities associated with these SARs, except ensuring the information is provided. The functional specification documentation is provided to support the evaluation activities described in Section 5.1 of the OS PP, and other activities described for AGD, ATE, and AVA SARs. The requirements on the content of the functional specification information is implicitly assessed by virtue of the other evaluation activities being performed; if the evaluator is unable to perform an activity because there is insufficient interface information, then an adequate functional specification has not been provided.

Evaluator Assessment:

The ST and TOE guidance documentation provides specification of the interfaces, and associated management functions in enough detail to perform the assurance activities that are specified

5.2 AGD: Guidance Documentation

5.2.1 Operational User Guidance (AGD_OPE.1)

5.2.1.1 Evaluation Activity:

Some of the contents of the operational guidance are verified by the evaluation activities in Section 5.1 Security Functional Requirements and evaluation of the OS according to the [CEM]. The following additional information is also required. If cryptographic functions are provided by the OS, the operational guidance ill contain instructions for configuring the cryptographic engine associated with the evaluated configuration of the OS. It will provide a warning to the administrator that use of other cryptographic engines was not evaluated nor tested during the CC evaluation of the OS. The documentation must describe the process for verifying updates to the OS by verifying a digital signature – this may be done by the OS or the underlying platform. The evaluator will verify that this process includes the following steps: Instructions for obtaining the update itself. This should include instructions for making the update accessible to the OS (e.g., placement in a specific directory). 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. The OS will likely contain security functionality that does not fall in the scope of evaluation under this PP. The operational guidance will make it clear to an administrator which security functionality is covered by the evaluation activities.

Evaluator Assessment:

Section 4 of the [AGD] provides instructions for configuring the TOE in the evaluated configuration. Section 6 of the [AGD] provides instructions for configuration of TLS in the evaluated configuration. Section 18 of the [AGD] describes the process for retrieving and applying updates to the OS as well as instructions to configure automatic updates for the system.

5.2.2 Preparative Procedures (AGD_PRE.1)

5.2.2.1 Evaluation Activity:

As indicated in the introduction above, there are significant expectations with respect to the documentation—especially when configuring the operational environment to support OS functional requirements. The evaluator shall check to ensure that the guidance provided for the OS adequately addresses all platforms claimed for the OS in the ST.

Evaluator Assessment:

The evaluator was able to successfully use the provided guidance documentation to configure the TOE in the evaluated configuration for all platforms claimed in the ST.

5.3 ALC: Life-Cycle Support

5.3.1 Labeling of the TOE (ALC_CMC.1)

The evaluator will check the ST to ensure that it contains an identifier (such as a product name/version number) that specifically identifies the version that meets the requirements of the ST. Further, the evaluator will check the AGD guidance and OS samples received for testing to ensure that the version number is consistent with that in the ST. If the vendor maintains a web site advertising the OS, the evaluator will examine the information on the web site to ensure that the information in the ST is sufficient to distinguish the product.

Evaluator Assessment:

Section 1.1 of the [ST] provides the identifier for TOE. The evaluator also verified that the TOE version referenced in the [AGD], [ST], and the TOE itself are consistent and sufficient to distinguish the product from other products.

5.3.2 TOE CM Coverage (ALC_CMS.1)

The "evaluation evidence required by the SARs" in this PP is limited to the information in the ST coupled with the guidance provided to administrators and users under the AGD requirements. By ensuring that the OS is specifically identified and that this identification is consistent in the ST and in the AGD guidance (as done in the evaluation activity for ALC_CMC.1), the evaluator implicitly confirms the information required by this component. Life-cycle support is targeted aspects of the developer's life-cycle and instructions to providers of applications for the developer's devices, rather than an in-depth examination of the TSF manufacturer's development and configuration management process. This is not meant to diminish the critical role that a developer's practices play in contributing to the overall trustworthiness of a product; rather, it's a reflection on the information to be made available for evaluation.

The evaluator will ensure that the developer has identified (in guidance documentation for application developers concerning the targeted platform) one or more development environments appropriate for use in developing

applications for the developer's platform. For each of these development environments, the developer will provide information on how to configure the environment to ensure that buffer overflow protection mechanisms in the environment(s) are invoked (e.g., compiler and linker flags). The evaluator will ensure that this documentation also includes an indication of whether such protections are on by default, or have to be specifically enabled. The evaluator will ensure that the TSF is uniquely identified (with respect to other products from the TSF vendor), and that documentation provided by the developer in association with the requirements in the ST is associated with the TSF using this unique identification.

Evaluator Assessment:

The evaluator has verified that the TOE matches the version listed in the [ST] and [AGD] documents.

Section 16 in the [AGD] provides information on how to configure the GCC compiler and linker to ensure that buffer overflow protection mechanisms in the environment are invoked. These flags must be specifically invoked by the developer when using the GCC compiler.

5.3.3 Timely Security Updates (ALC_TSU_EXT.1)

The evaluator will verify that the TSS contains a description of the timely security update process used by the developer to create and deploy security updates. The evaluator will verify that this description addresses the entire application. The evaluator will also verify that, in addition to the OS developer's process, any third-party processes are also addressed in the description. The evaluator will also verify that each mechanism for deployment of security updates is described.

The evaluator will verify that, for each deployment mechanism described for the update process, the TSS lists a time between public disclosure of a vulnerability and public availability of the security update to the OS patching this vulnerability, to include any third-party or carrier delays in deployment. The evaluator will verify that this time is expressed in a number or range of days.

The evaluator will verify that this description includes the publicly available mechanisms (including either an email address or website) for reporting security issues related to the OS. The evaluator will verify that the description of this mechanism includes a method for protecting the report either using a public key for encrypting email or a trusted channel for a website.

Evaluator Assessment:

The evaluator verified that the TSS entry for ALC_TSU_EXT.1 outlined in Section 7 of the [ST] provides a description of the timely security update process. The process in which security vulnerabilities can be reported to the developer is described in the TSS and web links are provided for these activities. Web links for published errata where users can track current vulnerabilities are also provided.

Oracle provides a mailing list that users can utilize for notification of security updates and an email address to submit security observations. Oracle also provides a public key to protect communication between the user and Oracle Security departments.

.

5.4 ATE: Tests

5.4.1 Independent Testing – Conformance (ATE_IND.1)

5.4.1.1 Evaluation Activity:

The evaluator will prepare a test plan and report documenting the testing aspects of the system, including any application crashes during testing. The evaluator will determine the root cause of any application crashes and include that information in the report. The test plan covers all of the testing actions contained in the [CEM] and the body of this PP's evaluation activities.

While it is not necessary to have one test case per test listed in an evaluation activity, the evaluator must document in the test plan that each applicable testing requirement in the ST is covered. The test plan identifies the platforms to be tested, and for those 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. The test plan describes the composition of each platform to be tested, and any setup that is 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 OS and its platform.

This also includes the configuration of the cryptographic engine to be used. The cryptographic algorithms implemented by this engine are those specified by this PP and used by the cryptographic protocols being evaluated (IPsec, TLS). The test plan identifies high-level test objectives as well as the test procedures to be followed to achieve those objectives. These procedures include expected results.

The test report (which could just be an annotated 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 will be a cumulative account, so if there was a test run that resulted in a failure; a fix installed; and then a successful re-run of the test, the report would show a "fail" and "pass" result (and the supporting details), and not just the "pass" result.

Evaluator Assessment:

The evaluator performed testing on the TOE in the evaluated configuration outlined in the [ST] to satisfy each of the test requirements of the PP. The test plan and results have been provided in the Evaluation Test Procedure document.

5.5 AVA: Vulnerability Assessment

5.5.1 Vulnerability Survey (AVA_VAN.1)

5.5.1.1 Evaluation Activity:

The evaluator will generate a report to document their findings with respect to this requirement. This report could physically be part of the overall test report mentioned in ATE_IND, or a separate document. The evaluator performs a search of public information to find vulnerabilities that have been found in similar applications with a particular focus on network protocols the application uses and document formats it parses. The evaluator documents the sources consulted and the vulnerabilities found in the report.

For each vulnerability found, the evaluator either provides a rationale with respect to its non-applicability, or the evaluator formulates a test (using the guidelines provided in ATE_IND) to confirm the vulnerability, if suitable. Suitability is determined by assessing the attack vector needed to take advantage of the vulnerability. If exploiting the vulnerability requires expert skills and an electron microscope, for instance, then a test would not be suitable and an appropriate justification would be formulated.

Evaluator Assessment:

The evaluator performed the following vulnerability assessment activities on the TOE in the evaluated configuration described in the [ST]:

- Full TCP and UDP port scans
- Automated vulnerability scans were performed
- Search of public domain information on CVEs related to the TOE and its included cryptographic components
- Search of the CISA Known Exploited Vulnerabilities Catalog for relevant vulnerabilities

It was determined that the TOE is not susceptible to any know vulnerabilities. Results of the vulnerability efforts have been provided in the Vulnerability Assessment Report.

6. Requirements for Functional Package for Transport Layer Security (TLS)

6.1 Security Functional Requirements

6.1.1 FCS_TLS_EXT.1.1

6.1.1.1 Guidance Documentation Assurance Activity

The evaluator shall ensure that the selections indicated in the ST are consistent with selections in the dependent components.

Evaluator Assessment:

The evaluator has confirmed that the [ST] selects TLS as a client for FCS_TLS_EXT.1. This is consistent with the selections of FCS_TLSC_EXT.1 TLS Client Protocol SFRs.

7. Appendix A – Optional Requirements

There are currently no Optional Requirements in the Package.

8. Appendix B – Selection-Based Requirements

8.1 FCS_TLSC_EXT.1 TLS Client Protocol

8.1.1 FCS_TLSC_EXT.1.1

8.1.1.1 TSS Assurance Activity

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

Evaluator Assessment:

Table 19 in Section 7 of the [ST] lists the supported TLS cipher suites. The listed cipher suites match those listed in the SFR.

8.1.1.2 Guidance Documentation Assurance Activity

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

Evaluator Assessment:

Section 6 - Configuring TLS in [AGD] provides instructions on configuration of the OS such that TLS conforms to the description in the TSS.

8.1.1.3 Test Assurance Activity

The evaluator shall also perform the following tests:

- **Test 1:** The evaluator shall establish a TLS connection using each of the cipher suites 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 EAP session. It is sufficient to observe the successful negotiation of a cipher suite 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 cipher suite being used (for example, that the cryptographic algorithm is 128-bit AES and not 256-bit AES).
- **Test 2:** The goal of the following test is to verify that the TOE accepts only certificates with appropriate values in the extendedKeyUsage extension, and implicitly that the TOE correctly parses the extendedKeyUsage extension as part of X.509v3 server certificate validation.

  The evaluator shall attempt to establish the connection using a server with a server certificate that contains the Server Authentication purpose in the extendedKeyUsage extension and verify that a connection is established. The evaluator shall repeat this test using a different, but otherwise valid and trusted, certificate that lacks the Server Authentication purpose in the extendedKeyUsage extension and ensure that a connection is not established. Ideally, the two certificates should be similar in structure, the types of identifiers used, and the chain of trust.
- **Test 3:** The evaluator shall send a server certificate in the TLS connection that does not match the server-

selected cipher suite (for example, send a ECDSA certificate while using the TLS_RSA_WITH_AES_128_CBC_SHA cipher suite or send a RSA certificate while using one of the ECDSA cipher suites.) The evaluator shall verify that the product disconnects after receiving the server's Certificate handshake message.

- **Test 4:** The evaluator shall configure the server to select the TLS_NULL_WITH_NULL_NULL cipher suite and verify that the client denies the connection.
- **Test 5:** The evaluator shall perform the following modifications to the traffic:
    - **Test 5.1:** Change the TLS version selected by the server in the Server Hello to an undefined TLS version (for example 1.5 represented by the two bytes 03 06) and verify that the client rejects the connection.
    - **Test 5.2:** Change the TLS version selected by the server in the Server Hello to the most recent unsupported TLS version (for example 1.1 represented by the two bytes 03 02) and verify that the client rejects the connection.
    - **Test 5.3:** [conditional] If DHE or ECDHE cipher suites are supported, modify at least one byte in the server's nonce in the Server Hello handshake message, and verify that the client does not complete the handshake and no application data flows.
    - **Test 5.4:** Modify the server's selected cipher suite in the Server Hello handshake message to be a cipher suite not presented in the Client Hello handshake message. The evaluator shall verify that the client does not complete the handshake and no application data flows.
    - **Test 5.5:** [conditional] If DHE or ECDHE cipher suites are supported, modify the signature block in the server's Key Exchange handshake message, and verify that the client does not complete the handshake and no application data flows. This test does not apply to cipher suites using RSA key exchange. If a TOE only supports RSA key exchange in conjunction with TLS, then this test shall be omitted.
    - **Test 5.6:** Modify a byte in the Server Finished handshake message, and verify that the client does not complete the handshake and no application data flows.
    - **Test 5.7:** Send a message consisting of random bytes from the server after the server has issued the Change Cipher Spec message and verify that the client does not complete the handshake and no application data flows. The message must still have a valid 5-byte record header in order to ensure the message will be parsed as TLS.

Evaluator Assessment:

Test 1: The evaluator initiated a connection to a remote TLS server and verified that the connection succeeds for each claimed cipher suite.

Test 2: The evaluator initiated a connection to a remote TLS server using a certificate with the Server Authentication purpose in the EKU field and verified the connection succeeds. An attempt to connect to the same TLS server using a certificate with a Client Authentication purpose in the EKU field failed.

Test 3: The evaluator attempted to connect to a remote TLS server using a certificate that did not match the selected cipher suite and verified the connection fails.

Test 4: The evaluator attempted to connect to a remote TLS server using the TLS_NULL_WITH_NULL_NULL cipher suited and verified the connection fails.

Test 5.1: The evaluator attempted to connect to a remote TLS server using an undefined TLS version and verified the connection fails.

Test 5.2: The evaluator attempted to connect to a remote TLS server using the most recent unsupported TLS version and verified the connection fails.

Test 5.3: The evaluator attempted to connect to a remote TLS server that modified the server nonce in the server hello message and verified the connection fails.

Test 5.4: The evaluator attempted to connect to a remote TLS server that modifies the cipher suite to a non-supported one and verified the connection fails.

Test 5.5: The evaluator attempted to connect to a remote TLS server that modifies the signature block on the server key exchange message and verified the connection fails.

Test 5.6: The evaluator attempted to connect to a remote TLS server that modifies the server finished message and verified the connection fails.

Test 5.7: The evaluator attempted to connect to a remote TLS server that sends a message consisting of random bytes after the change cipher spec message and verified the connection fails. The message retains a valid 5-byte record header.

8.1.2 FCS_TLSC_EXT.1.2

8.1.2.1 TSS Assurance Activity

The evaluator shall ensure that the TSS describes the client's method of establishing all reference identifiers from the 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 product.

Evaluator Assessment:

Table 19 in Section 7 of the [ST] describes the client's method of establishing all reference identifiers and describes the types of identifiers that are supported. The TOE does not support URI reference identifiers, SRV reference identifiers, or certificate pinning.

8.1.2.2 Guidance Documentation Assurance Activity

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.

Evaluator Assessment:

Section 22 in the [AGD] provides instructions for setting the reference identifier.

8.1.2.3 Test Assurance Activity [TD0499]

The evaluator shall configure the reference identifier according to the AGD guidance and perform the following tests during a TLS connection. If the TOE supports certificate pinning, all pinned certificates must be removed before performing Tests 1 through 6. A pinned certificate must be added prior to performing Test 7.

- **Test 1:** The evaluator shall present a server certificate that contains a CN that does not match the reference

identifier and does not contain the SAN extension. The evaluator shall verify that the connection fails.

Note that some systems might require the presence of the SAN extension. In this case the connection would still fail but for the reason of the missing SAN extension instead of the mismatch of CN and reference identifier. Both reasons are acceptable to pass Test 1.

- **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.
- **Test 3:** [conditional] If the TOE does not mandate the presence of the SAN extension, the evaluator shall present a server certificate that contains a CN that matches the reference identifier and does not contain the SAN extension. The evaluator shall verify that the connection succeeds. If the TOE does mandate the presence of the SAN extension, this Test shall be omitted.
- **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.
- **Test 5:** The evaluator shall perform the following wildcard tests with each supported type of reference identifier. The support for wildcards is intended to be optional. If wildcards are supported, the first, second, and third tests below shall be executed. If wildcards are not supported, then the fourth test below shall be executed.
    - **Test 5.1:** [conditional]: If wildcards are supported, 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.
    - **Test 5.2:** [conditional]: If wildcards are supported, the evaluator shall present a server certificate containing a wildcard in the left-most label but not preceding the public suffix (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.
    - **Test 5.3:** [conditional]: If wildcards are supported, the evaluator shall present a server certificate containing a wildcard in the left-most label immediately preceding the public suffix (e.g. *.com). The evaluator shall configure the reference identifier with a single left-most label (e.g. foo.com) and verify that the connection fails. The evaluator shall configure the reference identifier with two left-most labels (e.g. bar.foo.com) and verify that the connection fails.
    - **Test 5.4:** [conditional]: If wildcards are not supported, 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 fails.
- **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.
- **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.

Evaluator Assessment:

Test 1: The evaluator attempted to connect to a remote TLS server using a certificate containing an invalid CN and invalid SAN and verified the connection fails.

Test 2: The evaluator attempted to connect to a remote TLS server using a certificate with a valid CN and an invalid SAN and verified the connection fails.

Test 3: The evaluator attempted to connect to a remote TLS server using a certificate with a valid CN and no SAN extension and verified the connection fails.

Test 4: The evaluator attempted to connect to a remote TLS server using a certificate with an invalid CN and a valid SAN and verified the connection fails.

Test 5.1: The evaluator attempted to connect to a remote TLS server using a certificate with a wildcard not in the left-most label and verified the connection fails.

Test 5.2: The evaluator attempted to connect to a remote TLS server using a certificate with a wildcard in the left-most label with the reference identifier set to a single left-most label. The connection succeeds. The evaluator then configured the reference identifier without a left-most label and verified the connection fails. A final test with the reference identifier consisting of two left-most labels was verified to fail an attempted connection.

Test 5.3: The evaluator attempted to connect to a remote TLS server using a certificate with a wildcard preceding the public suffix and the reference identifier set to a single left-most label and verified the connection fails. The reference identifier was then configured with two left-most labels and the connection failed.

Test 5.4: N/A. Wildcards are supported by the TOE.

Test 6: N/A. URI and Service name reference identifiers are not supported.

Test 7: N/A. Certificate pinning is not supported.

8.1.3 FCS_TLSC_EXT.1.3

8.1.3.1 TSS Assurance Activity

If the selection for authorizing override of invalid certificates is made, then the evaluator shall ensure that the TSS includes a description of how and when user or administrator authorization is obtained. The evaluator shall also ensure that the TSS describes any mechanism for storing such authorizations, such that future presentation of such otherwise-invalid certificates permits establishment of a trusted channel without user or administrator action.

Evaluator Assessment:

FCS_TLSC_EXT.1.3 not make the selection “except when override is authorized” and will only establish a trusted channel if the server certificate is valid.

8.1.3.2 Test Assurance Activity [TD0513]

The evaluator shall demonstrate that using an invalid certificate (unless excepted) results in the function failing as

follows, unless excepted:

- **Test 1a**: The evaluator shall demonstrate that a server using a certificate with a valid certification path successfully connects.
- **Test 1b:** The evaluator shall modify the certificate chain used by the server in test 1a to be invalid and demonstrate that a server using a certificate without a valid certification path to a trust store element of the TOE results in an authentication failure.
- **Test 1c** [conditional]: If the TOE trust store can be managed, the evaluator shall modify the trust store element used in Test 1a to be untrusted and demonstrate that a connection attempt from the same server used in Test 1a results in an authentication failure.
- **Test 2:** The evaluator shall demonstrate that a server using a certificate which has been revoked results in an authentication failure.
- **Test 3:** The evaluator shall demonstrate that a server using a certificate which has passed its expiration date results in an authentication failure.
- **Test 4:** The evaluator shall demonstrate that a server using a certificate which does not have a valid identifier results in an authentication failure.

Evaluator Assessment:

Test 1a: The evaluator attempted to connect to a server using a certificate with a valid certificate path and verified the connection succeeds.

Test 1b: The evaluator modified the certificate chain in Test 1a to be invalid. The evaluator then attempted to connect to a server with the invalid certificate chain and verified that the connection fails.

Test 1c: This test is satisfied by the testing completed for FIA_X509_EXT.1 test #64.

Test 2: The evaluator attempted to connect to a server using a revoked certificate and verified that the connection failed.

Test 3: The evaluator attempted to connect to a server using a certificate that had expired and verified that the connection failed.

Test 4: This test is satisfied by FCS_TLSC_EXT1.2, Test #1.

8.2 FCS_TLSC_EXT.5 TLS Client Support for Supported Groups Extension

8.2.1 FCS_TLSC_EXT.5.1

8.2.1.1 TSS Assurance Activity

The evaluator shall verify that TSS describes the Supported Groups Extension.

Evaluator Assessment:

Table 19 in Section 7 of the [ST] describes the Supported Groups Extension. Supported groups are secp384r1 and secp521r.

8.2.1.2 Testing Assurance Activity

The evaluator shall also perform the following test:

- **Test 1:** The evaluator shall configure a server to perform key exchange using each of the TOE's supported curves and/or groups. The evaluator shall verify that the TOE successfully connects to the server.

Evaluator Assessment:

The evaluator connected to a TLS server using each of the TOE's supported group and verified the connection was successful.

9. Requirements for Functional Package for Secure Shell (SSH)

9.1 Security Requirements

9.1.1 FCS_SSH_EXT.1 SSH Protocol

9.1.1.1 TSS Assurance Activity

FCS_SSH_EXT.1.1

The evaluator shall ensure that the selections indicated in the ST are consistent with selections in this and subsequent components. Otherwise, this SFR is evaluated by activities for other SFRs

FCS_SSH_EXT.1.2

The evaluator shall check to ensure that the authentication methods listed in the TSS are identical to those listed in this SFR component; and, ensure if password-based authentication methods have been selected in the ST then these are also described; and, ensure that if keyboard-interactive is selected, it describes the multifactor authentication mechanisms provided by the TOE.

FCS_SSH_EXT.1.3

The evaluator shall check that the TSS describes how large packets are detected and handled.

FCS_SSH_EXT.1.4

The evaluator will check the description of the implementation of SSH in the TSS to ensure the encryption algorithms supported are specified. The evaluator will check the TSS to ensure that the encryption algorithms specified are identical to those listed for this component.

FCS_SSH_EXT.1.5

The evaluator will check the description of the implementation of SSH in the TSS to ensure the hashing algorithms supported are specified. The evaluator will check the TSS to ensure that the hashing algorithms specified are identical to those listed for this component.

FCS_SSH_EXT.1.6

The evaluator will check the description of the implementation of SSH in the TSS to ensure the shared secret establishment algorithms supported are specified. The evaluator will check the TSS to ensure that the shared secret establishment algorithms specified are identical to those listed for this component.

FCS_SSH_EXT.1.7

The evaluator will check the description of the implementation of SSH in the TSS to ensure the KDFs supported are specified. The evaluator will check the TSS to ensure that the KDFs specified are identical to those listed for this component.

FCS_SSH_EXT.1.8

The evaluator shall check the TSS to ensure that if the TOE enforces connection rekey or termination limits lower than the maximum values that these lower limits are identified.

In cases where hardware limitation will prevent reaching data transfer threshold in less than one hour, the evaluator shall check the TSS to ensure it contains:

a. An argument describing this hardware-based limitation and

b. Identification of the hardware components that form the basis of such argument.

For example, if specific Ethernet Controller or Wi-Fi radio chip is the root cause of such limitation, these subsystems shall be identified.

Evaluator Assessment:

FCS_SSH_EXT.1.1

The evaluator has verified that the selections indicated in the ST are consistent with selections in this and subsequent components.

The SSH software shall implement the SSH protocol that complies with RFCs 4251, 4252, 4253, 4254,4344,5656, 6668, and 8332 as a client and server.

FCS_SSH_EXT.1.2

The authentication methods available include password-based authentication and public key authentication. These authentication methods are identical to those selected in the ST.

FCS_SSH_EXT.1.3

Packets that exceed 262144 bytes are dropped at the application layer per RFC 4253. Once SSH packets are received, it is verified that it contains the packet length, padding length, payload and random padding. Once the packet information has been verified then the packet is decrypted. The packets are stored in a buffer. If the packet size is larger than permitted, the SSH packets are dropped, and the connection is terminated.

FCS_SSH_EXT.1.4

Section 7 of the [ST] specifies the encryption algorithms supported by implementation of SSH as follows: aes256-ctr (RFC 4344).

The algorithms are identical to those listed in the component.

FCS_SSH_EXT.1.5

Section 7 of the [ST] specifies support for the following hashing algorithms: hmac-sha2-256 (RFC 6668) and hmac-sha2-512 (RFC 6668). The list is identical to those listed for this component.

FCS_SSH_EXT.1.6

Section 7 of the [ST] specifies the key exchange algorithms:
ecdh-sha2-nistp384 (RFC 5656), ecdh-sha2-nistp521 (RFC 5656)

The algorithms specified are identical to those listed in the component.

FCS_SSH_EXT.1.7

Section 7 of the [ST] specifies the TOE uses SSH KDF as defined in RFC 4253 (Section 7.2), and RFC 5656 (Section 4) to derive the following cryptographic keys from a shared secret: session keys.

The KDFs specified are identical to those listed in the component.

FCS_SSH_EXT.1.8

Section 7 of the [ST] states TOE ensures that a rekey of the session keys occurs when any of the following thresholds are met: one hour connection time and no more than one gigabyte of transmitted data or no more than one gigabyte of received data.

9.1.1.2 Guidance Documentation Assurance Activity

FCS_SSH_EXT.1.1

None defined.

FCS_SSH_EXT.1.2

The evaluator shall check the guidance documentation to ensure the configuration options, if any, for authentication mechanisms provided by the TOE are described.

FCS_SSH_EXT.1.3

None defined.

FCS_SSH_EXT.1.4

The evaluator shall check the guidance documentation to ensure that it contains instructions to the administrator on

how to ensure that only the allowed mechanisms are used in SSH connections with the TOE.

FCS_SSH_EXT.1.5

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

FCS_SSH_EXT.1.6

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

FCS_SSH_EXT.1.7

None defined.

FCS_SSH_EXT.1.8

The evaluator shall check the guidance documentation to ensure that if the connection rekey or termination limits are configurable, it contains instructions to the administrator on how to configure the relevant connection rekey or termination limits for the TOE.

Evaluator Assessment:

FCS_SSH_EXT.1.2

Section 5.3 and Section 8 in the [AGD] provide configuration options for the authentication mechanisms provided by the TOE.

FCS_SSH_EXT.1.4

Section 5 in the [AGD] contains instructions on how to ensure that only the allowed mechanisms are used in SSH connections with the TOE.

FCS_SSH_EXT.1.5

Section 5 in the [AGD] contains instructions on how to ensure that only the allowed mechanisms are used in SSH connections with the TOE.

FCS_SSH_EXT.1.6

Section 5 in the [AGD] contains instructions on how to ensure that only the allowed mechanisms are used in SSH connections with the TOE.

FCS_SSH_EXT.1.8

Section 5.4 in the [AGD] provides instructions to configure the relevant connection rekey limits for the TOE.

9.1.1.3 Test Assurance Activity [TD0732]

FCS_SSH_EXT.1.1

There are no test evaluation activities for this component. This SFR is evaluated by activities for other SFRs.

Evaluator Assessment:

N/A

FCS_SSH_EXT.1.2

- **Test 1:** [conditional] If the TOE is acting as SSH Server:
    a. The evaluator shall use a suitable SSH Client to connect to the TOE, enable debug messages in the SSH Client, and examine the debug messages to determine that only the configured authentication methods for the TOE were offered by the server.
    b. [conditional] If the SSH server supports X509 based Client authentication options:
        a. The evaluator shall initiate an SSH session from a client where the username is associated with the X509 certificate. The evaluator shall verify the session is successfully established.
        b. Next the evaluator shall use the same X509 certificate as above but include a username not associated with the certificate. The evaluator shall verify that the session does not establish.
        c. Finally, the evaluator shall use the correct username (from step a above) but use a different X509 certificate which is not associated with the username. The evaluator shall verify that the session does not establish.
- **Test 2:** [conditional] If the TOE is acting as SSH Client, the evaluator shall test for a successful configuration setting of each authentication method as follows:
    a. The evaluator shall initiate a SSH session using the authentication method configured and verify that the session is successfully established.
    b. Next, the evaluator shall use bad authentication data (e.g. incorrectly generated certificate or incorrect password) and ensure that the connection is rejected.

    Steps a-b shall be repeated for each independently configurable authentication method supported by the server.

- **Test 3:** [conditional] If the TOE is acting as SSH Client, the evaluator shall verify that the connection fails upon configuration mismatch as follows:
    a. The evaluator shall configure the Client with an authentication method not supported by the Server.
    b. The evaluator shall verify that the connection fails.

If the Client supports only one authentication method, the evaluator can test this failure of connection by configuring the Server with an authentication method not supported by the Client. In order to facilitate this test, it is acceptable for the evaluator to configure an authentication method that is outside of the selections in the SFR.

Evaluator Assessment:

Test 1:

a. The evaluator used an SSH client and connected to the TOE with debug messages enabled. Only the configured authentication methods were available.
b. N/A the TOE does not support x509 based client authentication.

Test 2:

a. The evaluator made successful connections to an SSH server from the TOE using both password-based authentication and public key authentication methods.
b. The evaluator attempted to connect to an SSH sever with a bad password, and then also attempted to connect using an incorrect public key and verified that both attempts were rejected.

Test 3:

a. The evaluator configured the server to only support public-key authentication.
b. The evaluator attempted to connect using password authentication and verified the connection failed.

FCS_SSH_EXT.1.3

- **Test 1:** The evaluator shall demonstrate that the TOE accepts the maximum allowed packet size.
- **Test 2:** This test is performed to verify that the TOE drops packets that are larger than size specified in the component.
    a. The evaluator shall establish a successful SSH connection with the peer.
    b. Next the evaluator shall craft a packet that is slightly larger than the maximum size specified in this component and send it through the established SSH connection to the TOE. The packet should not be greater than the maximum packet size + 16 bytes. If the packet is larger, the evaluator shall justify the need to send a larger packet.
    c. The evaluator shall verify that the packet was dropped by the TOE. The method of verification will vary by the TOE. Examples include reviewing the TOE audit log for a dropped packet audit or observing the TOE terminates the connection.

Evaluator Assessment:

Test 1:

The evaluator used a tool to generate a packet of a specific size and send it to the TOE. The large packet is accepted by the TOE.

Test 2:

The evaluator used a tool to generate a packet larger than the maximum size allowed by the TOE and verified that the TOE dropped the packet and logged the action. The size of the packet sent was within the range of the maximum packet size + 16 bytes.

FCS_SSH_EXT.1.4

The evaluator shall perform the following tests.
If the TOE can be both a client and a server, these tests must be performed for both roles.

- **Test 1:** The evaluator must ensure that only claimed algorithms and cryptographic primitives are used to establish an SSH connection. To verify this, the evaluator shall establish an SSH connection with a remote endpoint. The evaluator shall capture the traffic exchanged between the TOE and the remote endpoint during protocol negotiation (e.g. using a packet capture tool or information provided by the endpoint, respectively). The evaluator shall verify from the captured traffic that the TOE offers only the algorithms defined in the ST for the TOE for SSH connections. The evaluator shall perform one successful negotiation of an SSH connection and verify that the negotiated algorithms were included in the advertised set. If the evaluator detects that not all algorithms defined in the ST for SSH are advertised by the TOE or the TOE advertises additional algorithms not defined in the ST for SSH, the test shall be regarded as failed.

  The data collected from the connection above shall be used for verification of the advertised hashing and shared secret establishment algorithms in FCS_SSH_EXT.1.5 and FCS_SSH_EXT.1.6 respectively.

- **Test 2:** For the connection established in Test 1, the evaluator shall terminate the connection and observe that the TOE terminates the connection.
- **Test 3:** The evaluator shall configure the remote endpoint to only allow a mechanism that is not included in the ST selection. The evaluator shall attempt to connect to the TOE and observe that the attempt fails.

Evaluator Assessment:

Test 1:

The evaluator established connections with the TOE acting both as a server and a client and captured the traffic with a packet capture tool. After making the connections the evaluator verified that only the algorithms defined in the ST were offered for the server and client respectively.

Test 2:

The evaluator terminated the connection established in Test 1 and observed that the connection is terminated successfully.

Test 3:

The evaluator modified the remote endpoint to only accept a mechanism not included in the ST selection and attempted to connection to the TOE. The attempt to connect failed.

FCS_SSH_EXT.1.5

- **Test 1:** The evaluator shall use the test data collected in FCS_SSH_EXT.1.4, Test 1 to verify that appropriate mechanisms are advertised.
- **Test 2:** The evaluator shall configure an SSH peer to allow only a hashing algorithm that is not included in the ST selection. The evaluator shall attempt to establish an SSH connection and observe that the connection is rejected.

Evaluator Assessment:

Test 1:

The evaluator used the data collected in FCS_SSH_EXT.1.4 and verified that the appropriate mechanisms are advertised.

Test 2:

The evaluator configured an SSH peer to only allow a hashing algorithm not included in the ST selection. The evaluator then attempted to establish a connection and verified that the connection failed.

FCS_SSH_EXT.1.6

- **Test 1:** The evaluator shall use the test data collected in FCS_SSH_EXT.1.4, Test 1 to verify that appropriate mechanisms are advertised.
- **Test 2:** The evaluator shall configure an SSH peer to allow only a key exchange method that is not included in the ST selection. The evaluator shall attempt to establish an SSH connection and observe that the connection is rejected.

Evaluator Assessment:

Test 1:

The evaluator used the data collected in FCS_SSH_EXT.1.4 and verified that the appropriate mechanisms are advertised.

Test 2:

The evaluator configured an SSH peer to only allow only a key exchange method not included in the ST selection. The evaluator then attempted to establish a connection and verified that the connection failed.

FCS_SSH_EXT.1.7

No test defined.

Evaluator Assessment:

N/A

FCS_SSH_EXT.1.8

The test harness needs to be configured so that its connection rekey or termination limits are greater than the limits supported by the TOE -- it is expected that the test harness should not be initiating the connection rekey or termination.

- **Test 1:** Establish an SSH connection. Wait until the identified connection rekey limit is met. Observed that a connection rekey or termination is initiated. This may require traffic to periodically be sent, or connection keep alive to be set, to ensure that the connection is not closed due to an idle timeout.
- **Test 2:** Establish an SSH connection. Transmit data from the TOE until the identified connection rekey or termination limit is met. Observe that a connection rekey or termination is initiated.
- **Test 3:** Establish an SSH connection. Send data to the TOE until the identified connection rekey limit or termination is met. Observe that a connection rekey or termination is initiated.

Evaluator Assessment:

Test 1:

The evaluator established an SSH connection and observed that the connection rekeyed after the specified interval.

Test 2:

The evaluator established an SSH connection and transmitted data from the TOE. It was observed that connection rekeyed after the limit was reached.

Test 3:

The evaluator established an SSH connection and transmitted data to the TOE. It was observed that connection rekeyed after the limit was reached.

9.2 FCS_SSHC_EXT.1 SSH Protocol - Client

9.2.1.1 Guidance Documentation Assurance Activity

The evaluator shall check the guidance documentation to ensure that it contains instructions to the administrator on

how to ensure that only the allowed mechanisms are used in SSH connections with the TOE.

Evaluator Assessment:

Section 5.2- Configuring SSH Client in [AGD] provides information on how to ensure only the allowed mechanisms are used in SSH connections with the TOE.

9.2.1.2 Test Assurance Activity

The evaluator shall perform the following tests:

- **Test 1:** [conditional] If using a local database by associating each host name with its corresponding public key, the evaluator shall configure the TOE with only a single host name and corresponding public key in the local database. The evaluator shall verify that the TOE can successfully connect to the host identified by the host name.
- **Test 2:** [conditional] If using a local database by associating each host name with its corresponding public key, the evaluator shall configure the TOE with only a single host name and non-corresponding public key in the local database. The evaluator shall verify that the TOE fails to connect to a host not identified by the host name.
- **Test 3:** [conditional] If using a local database by associating each host name with its corresponding public key, the evaluator shall try to connect to a host not configured in the local database. The evaluator shall verify that the TOE either fails to connect to a host identified by the host name or there is a prompt provided to store the public key in the local database.
- **Test 4:** [conditional] If using a list of trusted certification authorities, the evaluator shall configure the TOE with only a single trusted certification authority corresponding to the host. The evaluator shall verify that the TOE can successfully connect to the host identified by the host name.
- **Test 5:** [conditional] If using a list of trusted certification authorities, the evaluator shall configure the TOE with only a single trusted certification authority that does not correspond to the host. The evaluator shall verify that the TOE fails to the host identified by the host name.

Evaluator Assessment:

Test 1:

The evaluator verified that there is an entry in the local database containing the public key of the host. The evaluator then attempted to connect to the host and verified the connection was successful.

Test 2:

The evaluator configured the local database with a single hostname and non-corresponding public key. The evaluator then attempted to connect to the host and verified the TOE failed to connect to the host not identified by the host name.

Test 3:

The evaluator tried to connect to a host that was not configured in the local database. The connection to the server is not accepted and the user is prompted to store the public key in the local database.

Test 4:

N/A The TOE does not use a list of trusted certification authorities.

Test 5:

N/A The TOE does not use a list of trusted certification authorities.

9.3 FCS_SSHS_EXT.1 SSH Protocol - Server

9.3.1.1 Guidance Documentation Assurance Activity

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

Evaluator Assessment:

Section 5.1- Configuring SSH Server in [AGD] provides information on how to ensure only the allowed mechanisms are used in SSH connections with the TOE.

9.3.1.2 Test Assurance Activity [TD0682]

The evaluator shall perform the following tests:

Test 1: The evaluator shall use a suitable SSH Client to connect to the TOE and examine the list of server host key algorithms in the SSH_MSG_KEXINIT packet sent from the server to the client to determine that only the configured server authentication methods for the TOE were offered by the server.

Test 2: The evaluator shall test for a successful configuration setting of each server authentication method as follows. The evaluator shall initiate a SSH session using the authentication method configured and verify that the session is successfully established. Repeat this process for each independently configurable server authentication method supported by the server.

Test 3: The evaluator shall configure the peer to only allow an authentication mechanism that is not included in the ST selection. The evaluator shall attempt to connect to the TOE and observe that the TOE sends a disconnect message.

Evaluator Assessment:

Test 1:

The evaluator connected to the TOE with a suitable SSH client and verified that only the configured host key algorithms were offered by the TOE in the SSH_MSG_KEXINIT packet sent from the server to the client.

Test 2:

The evaluator initiated a SSH session using each of the configured authentication methods and verified that the session was successfully established.

Test 3:

The evaluator configured an SSH client to only allow an authentication method not included in the ST selection and attempted to connect to the TOE. The attempt to connect fails and the TOE sends a disconnect message to the client.