www.GossamerSec.com

Assurance Activity Report for Infinera Corporation Transcend Network Management System Server 18.10.3

Version 0.3
12/09/22

Prepared by:
Gossamer Security Solutions
Accredited Security Testing Laboratory – Common Criteria Testing
Columbia, MD 21045

Prepared for:
National Information Assurance Partnership
Common Criteria Evaluation and Validation Scheme

Revision History

Revision	Date	Authors	Summary
Version 0.1	11/11/22	Smoley	Initial draft
Version 0.2	12/07/22	Smoley	Updated for first-round comments
Version 0.3	12/09/22	Smoley	Updated for second-round comments

The TOE Evaluation was Sponsored by:

Infinera Corporation
9005 Junction Drive, Suite C
Annapolis Junction, MD 20701

Evaluation Personnel:

- Raymond Smoley

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

Table of Contents

1. Introduction 5
1.1 Test Equivalence 5
1.2 CAVP Certificates 5
2. Protection Profile SFR Assurance Activities 6
2.1 Cryptographic support (FCS) 6
2.1.1 Cryptographic Key Generation Services (ASPP14:FCS_CKM.1) 6
2.1.2 Cryptographic Asymmetric Key Generation - per TD0659 (ASPP14:FCS_CKM.1/AK) 6
2.1.3 Password Conditioning (ASPP14:FCS_CKM.1/PBKDF) 10
2.1.4 Cryptographic Symmetric Key Generation (ASPP14:FCS_CKM.1/SK) 11
2.1.5 Cryptographic Key Establishment (ASPP14:FCS_CKM.2) 12
2.1.6 Cryptographic Operation - Hashing (ASPP14:FCS_COP.1/Hash) 15
2.1.7 Cryptographic Operation - Keyed-Hash Message Authentication - per TD0626 (ASPP14:FCS_COP.1/KeyedHash) 17
2.1.8 Cryptographic Operation - Signing (ASPP14:FCS_COP.1/Sig) 17
2.1.9 Cryptographic Operation - Encryption/Decryption (ASPP14:FCS_COP.1/SKC) 18
2.1.10 Random Bit Generation Services (ASPP14:FCS_RBG_EXT.1) 25
2.1.11 Random Bit Generation from Application (ASPP14:FCS_RBG_EXT.2) 27
2.1.12 SSH Protocol (SSH10:FCS_SSH_EXT.1) 29
2.1.13 SSH Protocol - Client (SSH10:FCS_SSHC_EXT.1) 37
2.1.14 Storage of Credentials (ASPP14:FCS_STO_EXT.1) 39
2.1.15 TLS Protocol (PKGTLS11:FCS_TLS_EXT.1) 41
2.1.16 TLS Server Protocol (PKGTLS11:FCS_TLSS_EXT.1) 41
2.2 User data protection (FDP) 46
2.2.1 Encryption Of Sensitive Application Data (ASPP14:FDP_DAR_EXT.1) 46
2.2.2 Access to Platform Resources (ASPP14:FDP_DEC_EXT.1) 48
2.2.3 Network Communications (ASPP14:FDP_NET_EXT.1) 50
2.3 Security management (FMT) 51
2.3.1 Secure by Default Configuration (ASPP14:FMT_CFG_EXT.1) 51
2.3.2 Supported Configuration Mechanism - per TD0624 (ASPP14:FMT_MEC_EXT.1) 53
2.3.3 Specification of Management Functions (ASPP14:FMT_SMF.1) 55

2.4 Privacy (FPR) 56
2.4.1 User Consent for Transmission of Personally Identifiable (ASPP14:FPR_ANO_EXT.1) 56
2.5 Protection of the TSF (FPT) 57
2.5.1 Anti-Exploitation Capabilities (ASPP14:FPT_AEX_EXT.1) 57
2.5.2 Use of Supported Services and APIs (ASPP14:FPT_API_EXT.1) 62
2.5.3 Software Identification and Versions (ASPP14:FPT_IDV_EXT.1) 63
2.5.4 Use of Third Party Libraries (ASPP14:FPT_LIB_EXT.1) 63
2.5.5 Integrity for Installation and Update (ASPP14:FPT_TUD_EXT.1) 64
2.5.6 Integrity for Installation and Update - per TD0664 (ASPP14:FPT_TUD_EXT.2) 66
2.6 Trusted path/channels (FTP) 69
2.6.1 Protection of Data in Transit - per TD0655 (ASPP14:FTP_DIT_EXT.1) 69
3. Protection Profile SAR Assurance Activities 71
3.1 Development (ADV) 71
3.1.1 Basic Functional Specification (ADV_FSP.1) 71
3.2 Guidance documents (AGD) 71
3.2.1 Operational User Guidance (AGD_OPE.1) 71
3.2.2 Preparative Procedures (AGD_PRE.1) 72
3.3 Life-cycle support (ALC) 72
3.3.1 Labelling of the TOE (ALC_CMC.1) 72
3.3.2 TOE CM Coverage (ALC_CMS.1) 73
3.3.3 Timely Security Updates (ALC_TSU_EXT.1) 73
3.4 Tests (ATE) 74
3.4.1 Independent Testing - Conformance (ATE_IND.1) 74
3.5 Vulnerability assessment (AVA) 75
3.5.1 Vulnerability Survey (AVA_VAN.1) 75

1. Introduction

This document presents evaluations results of the Infinera Corporation Transcend Network Management System (TNMS) Server 18.10.3 ASPP14/SSH10/PKGTLS11 evaluation. This document contains a description of the assurance activities and associated results as performed by the evaluators.

1.1 Test Equivalence

The evaluator fully tested on the TNMS Server on a CentOS 7.9 platform. The TOE also claims compatibility for any RedHat/CentOS 7.9 deployment and multiple processors. Since the TOE is a Java application which abstracts the function calls made by the TOE and the same TOE image is provided across different deployments, the evaluation team concludes that the evidence provided should be sufficient to match all of the deployments claimed in the ST.

Additionally, the evaluator made use of an external TNMS Client and an MTERA network element to produce test evidence. The TNMS Client and MTERA products are not a part of this evaluation as they are separately evaluated, and are is only used to demonstrate the correct operations of the TOE. All security requirements are met by the TOE completely. Functional behavior from the TNMS client or any configured network elements are not claimed under this evaluation.

1.2 CAVP Certificates

The TOE performs cryptographic algorithms in accordance with the following NIST standards and has received the following CAVP algorithm certificates.

SFR	Algorithm	NIST Standard	Cert#
FCS_CKM.1/AK (Key Gen)	RSA IFC key generation 2048 bits	FIPS 186-4, RSA	A2313
	ECDSA ECC key gen P-256, 384, 521	FIPS 186-4, ECDSA	A2313
FCS_CKM.1/PDKDF (Password Conditioning)	PBKDFv2 (HMAC-SHA2-256)	SP 800-108	A2313
FCS_CKM.2 (Key Establishment)	ECC-based key exchange	SP 800-56A, KAS ECC	A2313
FCS_COP.1/SKC	AES Encryption/Decryption CBC, GCM, CTR 128, 256 bits	FIPS 197, SP 800-38	A2313
FCS_COP.1/Hash	SHA Hashing SHA-1, 256, 384, 512	FIPS 180-4	A2313
FCS_COP.1/Sig	RSA Sign/Verify 2048 bits	FIPS 186-4, RSA	A2313
	ECDSA Sign/Verify P-256, 384, 521	FIPS 186-4, ECDSA	A2313
FCS_COP.1/KeyedHash	HMAC-SHA HMAC-SHA 1, 256, 384, 512	FIPS 198-1 & 180-4	A2313
FCS_RBG_EXT.2 (Random)	DRBG Bit Generation Hash DRBG 256 bits	SP 800-90A	A2313

Table 1-1 Bouncy Castle CAVP Certificates

2. Protection Profile SFR Assurance Activities

This section of the AAR identifies each of the assurance activities included in the claimed Protection Profile and describes the findings in each case.

The following evidence was used to complete the Assurance Activities:

AAR v0.3

- Infinera Corporation Transcend Network Management System Server 18.10.3 Security Target, Version 1.5 12/09/2022 **[ST]**
- Infinera Transcend Network Management System Server 18.10.3 Administrative Guidance for Common Criteria, Version 1.1, December 6th 2022 **[Admin Guide]**

2.1 Cryptographic support (FCS)

2.1.1 Cryptographic Key Generation Services (ASPP14:FCS_CKM.1)

2.1.1.1 ASPP14:FCS_CKM.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall inspect the application and its developer documentation to determine if the application needs asymmetric key generation services. If not, the evaluator shall verify the generate no asymmetric cryptographic keys selection is present in the ST. Otherwise, the evaluation activities shall be performed as stated in the selection-based requirements.

Section 6.1 of the ST states that the TOE generates asymmetric ECDH keys during TLS and SSHv2 connections. The evaluator examined the AGD document and found references to algorithms used by TLS and SSH, however no other references to key generation outside of those claimed in the ST.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.1.2 Cryptographic Asymmetric Key Generation - per TD0659 (ASPP14:FCS_CKM.1/AK)

2.1.2.1 ASPP14:FCS_CKM.1/AK.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

If the application 'invokes platform-provided functionality for asymmetric key generation', then the evaluator shall examine the TSS to verify that it describes how the key generation functionality is invoked.

Section 6.1 of the ST states that the TOE performs the following uses for asymmetric key generation and their key sizes.

- Generates P-256 and P-384 EDCHE keys as part of TLS secured connections from incoming TNMS Clients (administrative traffic).
- The TOE also supports SSHv2 session establishment using ECDH (ecdh-sha2-nistp256, ecdh-sha2-nistp384, ecdh-sha2-nistp512). The TOE secures configuration connections to network devices under its administrative management control.

The TOE does not claim to invoke any platform-provided functionality for asymmetric key generation.

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

The TOE generates ECDH keys as a part of TLS secured administrative connections from incoming TNMS Clients and ECDH keys for SSHv2 connections to configured network elements.

For TLS, *Section 4.5 Managing Certificates and Keys* of the AGD contains information about configuring TLS server certificates, including the support for both ECDSA P-256 and RSA-2048 server certificates. Additionally, *Section 4.5.3 TLS Configuration* also states “Other than setting the trusted root certificates, TLS settings in TNMS Server – such as supported cipher suites and key sizes - are fixed and not configurable by the user.”

For SSH, *Section 4.16 DCN Management* of the AGD contains information about configuring SSH which is done on a per-network element basis. Included in this section are settings related to enable SSH (use secure connection), configure authentication type (PASSWORD or SSH_KEY), configuring PEM-based SSH private keys, and configuring SSH Host keys. *Section 4.16.2 SSH Algorithm Configuration* states “SSH algorithms in TNMS Client are fixed and non-configurable.” Lastly, *Section 4.16.3 Adding Network Elements to TNMS* spells out all of the supported SSH key algorithms.

Additionally, *Section 4.4 Enabling FIPS mode* of the AGD indicates that the TOE installation requires setting up the cryptographic library into FIPS mode in order to be in compliance with the Common Criteria evaluated configuration. Additionally, this section also states that “the use of any other cryptographic engines, or

configurations other than what is described in this section was not evaluated nor tested during the Common Criteria Evaluation of TNMS Server.

Component Testing Assurance Activities: If the application 'implements asymmetric key generation,' then the following test activities shall be carried out. Evaluation Activity Note: The following tests may require the developer to provide access to a developer environment that provides the evaluator with tools that are typically available to end-users of the application. Key Generation for FIPS PUB 186-4 RSA Schemes The evaluator shall verify the implementation of RSA Key Generation by the TOE using the Key Generation test. This test verifies the ability of the TSF to correctly produce values for the key components including the public verification exponent e, the private prime factors p and q, the public modulus n and the calculation of the private signature exponent d. 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 shall 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 shall 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,

2^16 <= e <= 2^256 and e is an odd integer,

|p-q| > 2^(nlen/2 - 100),

p >= 2^(nlen/2 -1/2),

q >= 2^(nlen/2 -1/2),

2^(nlen/2) < d < LCM(p-1,q-1),

e*d = 1 mod LCM(p-1,q-1).

Key Generation for Elliptic Curve Cryptography (ECC)

FIPS 186-4 ECC Key Generation Test For each supported NIST curve, i.e., P-256, P384 and P-521, the evaluator shall require the implementation under test (IUT) to generate 10 private/public key pairs. The private key shall be generated using an approved random bit generator (RBG). To determine correctness, the evaluator shall submit the generated key pairs to the public key verification (PKV) function of a known good implementation. FIPS 186-4 Public Key Verification (PKV) Test For each supported NIST curve, i.e., P256, P-384 and P-521, the evaluator shall generate 10 private/public key pairs using the key generation function of a known good implementation and modify five of the public key values so that they are incorrect, leaving five values unchanged (i.e., correct). The evaluator shall obtain in response a set of 10 PASS/FAIL values. Key Generation for Finite-Field Cryptography (FFC) The evaluator shall verify the implementation of the Parameters Generation and the Key Generation for FFC by the TOE using the Parameter Generation and Key Generation test. This test verifies the ability of the TSF to correctly produce values for the field prime p, the cryptographic prime q (dividing p-1), the cryptographic group generator g, and the calculation of the private key x and public key y. 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 shall have the TSF generate 25 parameter sets and key pairs. The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated from a known good implementation. Verification must also confirm

g not= 0,1

q divides p-1

g^q mod p = 1

g^x mod p = y

for each FFC parameter set and key pair.

Diffie-Hellman Group 14 and FFC Schemes using 'safe-prime' groups

Testing for FFC Schemes using Diffie-Hellman group 14 and/or safe-prime groups is done as part of testing in CKM.2.1.

The TOE TNMS Server includes the CAVP-certified BC-FJA (Bouncy Castle FIPS Java API). See Section 1.2 for a listing of CAVP certificates.

2.1.3 Password Conditioning (ASPP14:FCS_CKM.1/PBKDF)

2.1.3.1 ASPP14:FCS_CKM.1/PBKDF.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.3.2 ASPP14:FCS_CKM.1/PBKDF.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: Support for PBKDF: The evaluator shall examine the password hierarchy TSS to ensure that the formation of all password based derived keys is described and that the key sizes match that described by the ST author. The evaluator shall check that the TSS describes the method by which the password/passphrase is first encoded and then fed to the SHA algorithm. The settings for the algorithm (padding,

blocking, etc.) shall be described, and the evaluator shall verify that these are supported by the selections in this component as well as the selections concerning the hash function itself. The evaluator shall verify that the TSS contains a description of how the output of the hash function is used to form the submask that will be input into the function. For the NIST SP 800-132-based conditioning of the password/passphrase, the required evaluation activities will be performed when doing the evaluation activities for the appropriate requirements (FCS_COP.1.1/KeyedHash). No explicit testing of the formation of the submask from the input password is required. FCS_CKM.1.1/PBKDF: The ST author shall provide a description in the TSS regarding the salt generation. The evaluator shall confirm that the salt is generated using an RBG described in FCS_RBG_EXT.1.

Section 6.1 of the ST states that PBKDF is utilized with TLS protected Client connections as follows:

> The TOE receives a SHA-1 hash value derived from the administrator's password. The TOE then uses that binary value, appends a random 16 byte salt, and then subjects the concatenation to PBKDFv2 (HMAC-SHA-256) conditioning to generate a 128-bit output. The TOE generates the per-user 16-byte salt using its Bouncy Castle DRBG.

The hash functions and RBG (further explored under FCS_RBG_EXT.1) used under PBKDF are CAVP certified. See Section 1.2 for a listing of CAVP certificates.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.1.4 Cryptographic Symmetric Key Generation (ASPP14:FCS_CKM.1/SK)

2.1.4.1 ASPP14:FCS_CKM.1/SK.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall review the TSS to determine that it describes how the functionality described by FCS_RBG_EXT.1 is invoked.

If the application is relying on random bit generation from the host platform, the evaluator shall verify the TSS includes the name/manufacturer of the external RBG and describes the function call and parameters used when calling the external DRBG function. If different external RBGs are used for different platforms, the evaluator shall verify the TSS identifies each RBG for each platform. Also, the evaluator shall verify the TSS includes a short description of the vendor's assumption for the amount of entropy seeding the external DRBG. The evaluator uses the description of the RBG functionality in FCS_RBG_EXT or documentation available for the operational environment to determine that the key size being requested is identical to the key size and mode to be used for the encryption/decryption of the user data.

Section 6.1 of the ST states that the TOE uses AES symmetric key generation as a part of TLS and SSH. Both of these keys are generated with the CAVP certified Bouncy Castle library. Under this section the ST states the TOE's Bouncy Castle library calls the java.security.SecureRandom class [specifically calling SecureRandom.generateSeed()] to obtain a 256-bit seed, which is assumed to contain 256-bits of entropy and additionally "the TOE obtains entropy to seed its DRBG from the platform through the /dev/urandom character device.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.1.5 Cryptographic Key Establishment (ASPP14:FCS_CKM.2)

2.1.5.1 ASPP14:FCS_CKM.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.1 of the ST states the TOE uses ECDHE as part of its TLS connection from TNMS Clients, and the TOE additionally uses ECDH in SSHv2 key exchange for to TOE connections to network devices that the TOE manages. Those claims match the key generation schemes identified under FCS_CKM.1.1.

Component Guidance Assurance Activities: The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected key establishment scheme(s).

The TOE uses ECDHE as a part of TLS secured administrative connections from incoming TNMS Clients and ECDHE for SSHv2 connections to configured network elements.

For TLS, *Section 4.5 Managing Certificates and Keys* of the AGD contains information about configuring TLS server certificates, including the support for both ECDSA P-256 and RSA-2048 server certificates. Additionally, *Section 4.5.3 TLS Configuration* also states "Other than setting the trusted root certificates, TLS settings in TNMS Server – such as supported cipher suites and key sizes - are fixed and not configurable by the user."

For SSH, *Section 4.16 DCN Management* of the AGD contains information about configuring SSH which is done on a per-network element basis. Included in this section are settings related to enable SSH (use secure connection), configure authentication type (PASSWORD or SSH_KEY), configuring PEM-based SSH private keys, and configuring SSH Host keys. *Section 4.16.2 SSH Algorithm Configuration* states "SSH algorithms in TNMS Client are fixed and non-configurable." Lastly, *Section 4.16.3 Adding Network Elements to TNMS* spells out all of the supported SSH key algorithms.

Additionally, *Section 4.4 Enabling FIPS mode* of the AGD indicates that the TOE installation requires setting up the cryptographic library into FIPS mode in order to be in compliance with the Common Criteria evaluated configuration. Additionally, this section also states that the use of any other cryptographic engines, or configurations other than what is described in this section was not evaluated nor tested during the Common Criteria Evaluation of TNMS Server.

Component Testing Assurance Activities: 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 shall verify the implementation of the key establishment schemes supported by the TOE using the applicable tests below.

SP800-56A Key Establishment Schemes

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

Function Test

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

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

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

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

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

Validity Test

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

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

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

SP800-56B Key Establishment Schemes

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

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

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

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

To conduct this test the evaluator shall generate or obtain test vectors from a known good implementation of the TOE supported schemes. For each combination of supported key establishment scheme and its options (with our

without key confirmation if supported, for each supported key confirmation MAC function if key confirmation is supported, and for each supported mask generation function if KTS-OAEP is supported), the tester shall generate 10 sets of test vectors. Each test vector shall include the RSA private key, the plaintext keying material (KeyData), any additional input parameters if applicable, the MacTag in cases where key confirmation is incorporated, and the outputted ciphertext. For each test vector, the evaluator shall perform the key establishment decryption operation on the TOE and ensure that the outputted plaintext keying material (KeyData) is equivalent to the plaintext keying material in the test vector. In cases where key confirmation is incorporated, the evaluator shall perform the key confirmation steps and ensure that the outputted MacTag is equivalent to the MacTag in the test vector.

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

RSA-based key establishment

The evaluator shall verify the correctness of the TSF's implementation of RSAESPKCS1-v1_5 by using a known good implementation for each protocol selected in FTP_DIT_EXT.1 that uses RSAES-PKCS1-v1_5.

Diffie-Hellman Group 14

The evaluator shall verify the correctness of the TSF's implementation of Diffie-Hellman group 14 by using a known good implementation for each protocol selected in FTP_DIT_EXT.1 that uses Diffie-Hellman group 14.

FFC Schemes using 'safe-prime' groups

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_DIT_EXT.1 that uses safe-prime groups. This test must be performed for each safe-prime group that each protocol uses.

The TOE TNMS Server includes the CAVP-certified BC-FJA (Bouncy Castle FIPS Java API). See Section 1.2 for a listing of CAVP certificates.

2.1.6 Cryptographic Operation - Hashing (ASPP14:FCS_COP.1/Hash)

2.1.6.1 ASPP14:FCS_COP.1/Hash.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall 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.

Section 6.1 of the ST states the TOE uses SHA-1, SHA-256, SHA-384, and SHA-512 hashing when generating TLS server authentication signatures and while verifying SSHv2 pubkey signatures received from managed network devices. The TOE also uses SHA-256 during PBKDFv2 transformation of administrator passwords.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: 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 hashes only messages that are an integral number of bytes in length; i.e., the length (in bits) of the message to be hashed is divisible by 8. The second mode is the bit-oriented mode. In this mode the TSF hashes messages of arbitrary length. As there are different tests for each mode, an indication is given in the following sections for the bit-oriented vs. the byte-oriented testmacs. The evaluator shall perform all of the following tests for each hash algorithm implemented by the TSF and used to satisfy the requirements of this PP.

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 1: Short Messages Test - Bit oriented Mode The evaluators devise an input set consisting of m+1 messages, where m is the block length of the hash algorithm. The length of the messages range sequentially from 0 to m bits. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Test 2: Short Messages Test - Byte oriented Mode The evaluators devise an input set consisting of m/8+1 messages, where m is the block length of the hash algorithm. The length of the messages range sequentially from 0 to m/8 bytes, with each message being an integral number of bytes. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Test 3: Selected Long Messages Test - Bit oriented Mode The evaluators devise 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*i, where 1 <= i <= m. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Test 4: Selected Long Messages Test - Byte oriented Mode The evaluators devise 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*99*i, where 1 <= i <= m/8. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Test 5: Pseudorandomly Generated Messages Test This test is for byte-oriented implementations only. The evaluators randomly generate a seed that is n bits long, where n is the length of the message digest produced by the hash function to be tested. The evaluators then formulate a set of 100 messages and associated digests by following the algorithm provided in Figure 1 of [SHAVS]. The evaluators then ensure that the correct result is produced when the messages are provided to the TSF.

The TOE TNMS Server includes the CAVP-certified BC-FJA (Bouncy Castle FIPS Java API). See Section 1.2 for a listing of CAVP certificates.

2.1.7 Cryptographic Operation - Keyed-Hash Message Authentication - per TD0626 (ASPP14:FCS_COP.1/KeyedHash)

2.1.7.1 ASPP14:FCS_COP.1/KeyedHash.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

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

The TOE TNMS Server includes the CAVP-certified BC-FJA (Bouncy Castle FIPS Java API). See Section 1.2 for a listing of CAVP certificates.

2.1.8 Cryptographic Operation - Signing (ASPP14:FCS_COP.1/Sig)

2.1.8.1 ASPP14:FCS_COP.1/Sig.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The evaluator shall perform the following activities based on the selections in the ST.

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.

ECDSA Algorithm Tests

Test 1: ECDSA FIPS 186-4 Signature Generation Test. For each supported NIST curve (i.e., P-256, P-384 and P-521) and SHA function pair, the evaluator shall generate 10 1024-bit long messages and obtain for each message a public key and the resulting signature values R and S. To determine correctness, the evaluator shall use the signature verification function of a known good implementation.

Test 2: ECDSA FIPS 186-4 Signature Verification Test. For each supported NIST curve (i.e., P-256, P-384 and P-521) and SHA function pair, the evaluator shall generate a set of 10 1024-bit message, public key and signature tuples and modify one of the values (message, public key or signature) in five of the 10 tuples. The evaluator shall obtain in response a set of 10 PASS/FAIL values.

RSA Signature Algorithm Tests

Test 1: Signature Generation Test. The evaluator shall verify the implementation of RSA Signature Generation by the TOE using the Signature Generation Test. To conduct this test the evaluator must generate or obtain 10 messages from a trusted reference implementation for each modulus size/SHA combination supported by the TSF. The evaluator shall have the TOE use their private key and modulus value to sign these messages. The evaluator shall verify the correctness of the TSF's signature using a known good implementation and the associated public keys to verify the signatures.

Test 2: Signature Verification Test. The evaluator shall perform the Signature Verification test to verify the ability of the TOE to recognize another party's valid and invalid signatures. The evaluator shall inject errors into the test vectors produced during the Signature Verification Test by introducing errors in some of the public keys, e, messages, IR format, and/or signatures. The TOE attempts to verify the signatures and returns success or failure.

The TOE TNMS Server includes the CAVP-certified BC-FJA (Bouncy Castle FIPS Java API). See Section 1.2 for a listing of CAVP certificates.

2.1.9 Cryptographic Operation - Encryption/Decryption (ASPP14:FCS_COP.1/SKC)

2.1.9.1 ASPP14:FCS_COP.1/SKC.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: The evaluator checks the AGD documents to determine that any configuration that is required to be done to configure the functionality for the required modes and key sizes is present.

The TOE can negotiate both AES-CBC and AES-CTR ciphers as part of an SSHv2 connections and negotiate AES-GCM as part of a TLS connection.

For TLS, *Section 4.5.3 TLS Configuration* of the AGD states "Other than setting the trusted root certificates, TLS settings in TNMS Server – such as supported cipher suites and key sizes - are fixed and not configurable by the user."

For SSH, *Section 4.16 DCN Management* of the AGD contains information about configuring SSH which is done on a per-network element basis. *Section 4.16.2 SSH Algorithm Configuration* states "SSH algorithms in TNMS Client are fixed and non-configurable." Lastly, *Section 4.16.3 Adding Network Elements to TNMS* spells out all of the supported SSH key algorithms.

Additionally, *Section 4.4 Enabling FIPS mode* of the AGD indicates that the TOE installation requires setting up the cryptographic library into FIPS mode in order to be in compliance with the Common Criteria evaluated configuration. Additionally, this section also states that the use of any other cryptographic engines, or configurations other than what is described in this section was not evaluated nor tested during the Common Criteria Evaluation of TNMS Server.

Component Testing Assurance Activities: The evaluator shall perform all of the following tests for each algorithm implemented by the TSF and used to satisfy the requirements of this PP:

AES-CBC Known Answer Tests

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

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

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

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

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

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

AES-CBC Multi-Block Message Test

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

AES-CBC Monte Carlo Tests

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

Input: PT, IV, Key

for i = 1 to 1000:

if i == 1:

CT[1] = AES-CBC-Encrypt(Key, IV, PT)

PT = IV

else:

CT[i] = AES-CBC-Encrypt(Key, PT)

PT = CT[i-1]

The ciphertext computed in the 1000th iteration (i.e., CT[1000]) is the result for that trial. This result shall be compared to the result of running 1000 iterations with the same values using a known good implementation. The evaluator shall test the decrypt functionality using the same test as for encrypt, exchanging CT and PT and replacing AES-CBC-Encrypt with AES-CBC-Decrypt.

AES-GCM Monte Carlo Tests

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

128 bit and 256 bit keys

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

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

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

The evaluator shall test the encrypt functionality using a set of 10 key, plaintext, AAD, and IV tuples for each combination of parameter lengths above and obtain the ciphertext value and tag that results from AES-GCM authenticated encrypt. Each supported tag length shall be tested at least once per set of 10. The IV value may be supplied by the evaluator or the implementation being tested, as long as it is known.

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

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

AES-XTS Tests

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

256 bit (for AES-128) and 512 bit (for AES-256) keys

Three data unit (i.e., plaintext) lengths. One of the data unit lengths shall be a non-zero integer multiple of 128 bits, if supported. One of the data unit lengths shall be an integer multiple of 128 bits, if supported. The third data unit length shall be either the longest supported data unit length or 216 bits, whichever is smaller.

Using a set of 100 (key, plaintext and 128-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 shall 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 XTS-AES decrypt.

AES-CCM Tests

It is not recommended that evaluators use values obtained from static sources such as http://csrc.nist.gov/groups/STM/cavp/documents/mac/ccmtestvectors.zip or use values not generated expressly to exercise the AES-CCM implementation.

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

Keys: All supported and selected key sizes (e.g., 128, 256 bits).

Associated Data: Two or three values for associated data length: The minimum (. 0 bytes) and maximum (. 32 bytes) supported associated data lengths, and 2^16 (65536) bytes, if supported.

Payload: Two values for payload length: The minimum (. 0 bytes) and maximum (. 32 bytes) supported payload lengths.

Nonces: All supported nonce lengths (7, 8, 9, 10, 11, 12, 13) in bytes.

Tag: All supported tag lengths (4, 6, 8, 10, 12, 14, 16) in bytes.

The testing for CCM consists of five tests. To determine correctness in each of the below tests, the evaluator shall compare the ciphertext with the result of encryption of the same inputs with a known good implementation.

Variable Associated Data Test

For each supported key size and associated data length, and any supported payload length, nonce length, and tag length, the evaluator shall supply one key value, one nonce value, and 10 pairs of associated data and payload values, and obtain the resulting ciphertext.

Variable Payload Test

For each supported key size and payload length, and any supported associated data length, nonce length, and tag length, the evaluator shall supply one key value, one nonce value, and 10 pairs of associated data and payload values, and obtain the resulting ciphertext.

Variable Nonce Test

For each supported key size and nonce length, and any supported associated data length, payload length, and tag length, the evaluator shall supply one key value, one nonce value, and 10 pairs of associated data and payload values, and obtain the resulting ciphertext.

Variable Tag Test

For each supported key size and tag length, and any supported associated data length, payload length, and nonce length, the evaluator shall supply one key value, one nonce value, and 10 pairs of associated data and payload values, and obtain the resulting ciphertext.

Decryption-Verification Process Test

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 shall supply a key value and 15 sets of input plus ciphertext, and obtain the decrypted payload. Ten of the 15 input sets supplied should fail verification and five should pass.

AES-CTR Tests

Test 1: Known Answer Tests (KATs)

There are four Known Answer Tests (KATs) described below. For all KATs, the plaintext, IV, and ciphertext values shall be 128-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 shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

To test the encrypt functionality, the evaluator shall supply a set of 10 plaintext values and obtain the ciphertext value that results from encryption of the given plaintext using a key value of all zeros and an IV of all zeros. Five plaintext values shall be encrypted with a 128-bit all zeros key, and the other five shall be encrypted with a 256-bit all zeros key. To test the decrypt functionality, the evaluator shall perform the same test as for encrypt, using 10 ciphertext values as input.

To test the encrypt functionality, the evaluator shall supply a set of 10 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. Five of the key values shall be 128-bit keys, and the other five shall be 256-bit keys. To test the decrypt functionality, the evaluator shall perform the same test as for encrypt, using an all zero ciphertext value as input.

To test the encrypt functionality, the evaluator shall supply the two sets 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 an an IV of all zeros. The first set of keys shall have 128 128-bit keys, and the second shall have 256 256-bit keys. Key_i in each set shall have the leftmost i bits be ones and the rightmost N-i bits be zeros, for i in [1, N]. To test the decrypt functionality, the evaluator shall supply the two sets 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 first set of key/ciphertext pairs shall have 128 128-bit key/ciphertext pairs, and the second set of key/ciphertext pairs shall have 256 256-bit pairs. Key_i in each set shall have the leftmost i bits be ones and the rightmost N-i bits be zeros for i in [1, N]. The ciphertext value in each pair shall be the value that results in an all zeros plaintext when decrypted with its corresponding key.

To test the encrypt functionality, the evaluator shall supply the set of 128 plaintext values described below and obtain the two ciphertext values that result from encryption of the given plaintext using a 128-bit key value of all zeros and 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 128-i bits be zeros, for i in [1, 128]. To test the decrypt functionality, the evaluator shall perform the same test as for encrypt, using ciphertext values of the same form as the plaintext in the encrypt test as input.

Test 2: Multi-Block Message Test

The evaluator shall 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 shall 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 shall 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 shall 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.

Test 3: 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 shall test the encrypt functionality using 200 plaintext/key pairs. 100 of these shall use 128 bit keys, and 100 of these shall use 256 bit keys. The plaintext values shall be 128-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.

The TOE TNMS Server includes the CAVP-certified BC-FJA (Bouncy Castle FIPS Java API). See Section 1.2 for a listing of CAVP certificates.

2.1.10 Random Bit Generation Services (ASPP14:FCS_RBG_EXT.1)

2.1.10.1 ASPP14:FCS_RBG_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: If 'use no DRBG functionality' is selected, the evaluator shall inspect the application and its developer documentation and verify that the application needs no random bit generation services.

If 'implement DRBG functionality' is selected, the evaluator shall ensure that additional FCS_RBG_EXT.2 elements are included in the ST.

If 'invoke platform-provided DRBG functionality' is selected, the evaluator performs the following activities.

The evaluator shall examine the TSS to confirm that it identifies all functions (as described by the SFRs included in the ST) that obtain random numbers from the platform RBG. The evaluator shall determine that for each of these functions, the TSS states which platform interface (API) is used to obtain the random numbers. The evaluator shall confirm that each of these interfaces corresponds to the acceptable interfaces listed for each platform below.

It should be noted that there is no expectation that the evaluators attempt to confirm that the APIs are being used correctly for the functions identified in the TSS; the activity is to list the used APIs and then do an existence check via decompilation.

The ST claims that the TOE both invokes the platform-provided DRBG functionality and implements DRBG functionality as Section 6.1 states that the TOE invokes the platforms-provided DRBG functionality in order to obtain seeding material for its DRBG. As a result, the additional SFR FCS_RBG_EXT.2 was included in the ST.

The evaluator found the following quote under Section 6.1 that describes the functions this is used for: The TOE implements a DRBG in its Bouncy Castle library and uses that DRBG when generating per-user salts as well as when generating random values used in TLS handshakes, PKBDF credential transformation, and SSHv2 sessions.

Further information is shared under FCS_RBG_EXT.2 under Section 6.1 of the ST including the API used as the ST states the TOE obtains entropy to seed its DRBG from the platform through the /dev/random character device.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: If 'invoke platform-provided DRBG functionality' is selected, the following tests shall be performed:

The evaluator shall decompile the application binary using a decompiler suitable for the application (TOE). The evaluator shall search the output of the decompiler to determine that, for each API listed in the TSS, that API appears in the output. If the representation of the API does not correspond directly to the strings in the following list, the evaluator shall provide a mapping from the decompiled text to its corresponding API, with a description of why the API text does not directly correspond to the decompiled text and justification that the decompiled text corresponds to the associated API.

The following are the per-platform list of acceptable APIs:

Platforms: Android....

The evaluator shall verify that the application uses at least one of javax.crypto.KeyGenerator class or the java.security.SecureRandom class or /dev/random or /dev/urandom.

Platforms: Microsoft Windows....

The evaluator shall verify that rand_s, RtlGenRandom, BCryptGenRandom, or CryptGenRandom API is used for classic desktop applications. The evaluator shall verify the application uses the RNGCryptoServiceProvider class or derives a class from System.Security.Cryptography.RandomNumberGenerator API for Windows Universal Applications. It is only required that the API is called/invoked, there is no requirement that the API be used directly. In future versions of this document, CryptGenRandom may be removed as an option as it is no longer the preferred API per vendor documentation.

Platforms: Apple iOS....

The evaluator shall verify that the application invokes either SecRandomCopyBytes, CCRandomGenerateBytes or CCRandomCopyBytes, or uses /dev/random directly to acquire random.

Platforms: Linux....

The evaluator shall verify that the application collects random from /dev/random or /dev/urandom.

Platforms: Oracle Solaris....

The evaluator shall verify that the application invokes either CCRandomGenerateBytes or CCRandomCopyBytes, or collects random from /dev/random.

Platforms: Apple macOS....

The evaluator shall verify that the application invokes either CCRandomGenerateBytes or CCRandomCopyBytes, or collects random from /dev/random.

If invocation of platform-provided functionality is achieved in another way, the evaluator shall ensure the TSS describes how this is carried out, and how it is equivalent to the methods listed here (e.g. higher-level API invokes identical low-level API).

The evaluator decompiled the cryptographic library bundled with the TOE application as this was the component that was making cryptographic use of the randomization functions. The evaluator found several references to the standard Java API, SecureRandom. The evaluator found that by default, the Java Virtual Machine that was installed outside of the TOE boundary was configured to use /dev/random as the securerandom.source based on the java.security file, however the running processes that relate to the TOE all showed command line arguments passed int o java that overwrote this including "-Djava.security.egd=file:///dev/urandom". As a result, the TOE was shown to leverage /dev/urandom for its cryptographic seeding of its cryptographic library.

2.1.11 Random Bit Generation from Application (ASPP14:FCS_RBG_EXT.2)

2.1.11.1 ASPP14:FCS_RBG_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator shall perform the following tests, depending on the standard to which the RBG conforms. Implementations Conforming to FIPS 140-2 Annex C. The reference for the tests contained in this section is The Random Number Generator Validation System (RNGVS). The evaluators shall conduct the following two tests. Note that the 'expected values' are produced by a reference implementation of the algorithm that is known to be correct. Proof of correctness is left to each Scheme.

Test 1: The evaluators shall perform a Variable Seed Test. The evaluators shall provide a set of 128 (Seed, DT) pairs to the TSF RBG function, each 128 bits. The evaluators shall also provide a key (of the length appropriate to the AES algorithm) that is constant for all 128 (Seed, DT) pairs. The DT value is incremented by 1 for each set. The seed values shall have no repeats within the set. The evaluators ensure that the values returned by the TSF match the expected values.

Test 2: The evaluators shall perform a Monte Carlo Test. For this test, they supply an initial Seed and DT value to the TSF RBG function; each of these is 128 bits. The evaluators shall also provide a key (of the length appropriate to the AES algorithm) that is constant throughout the test. The evaluators then invoke the TSF RBG 10,000 times, with the DT value being incremented by 1 on each iteration, and the new seed for the subsequent iteration produced as specified in NISTRecommended Random Number Generator Based on ANSI X9.31 Appendix A.2.4 Using the 3-Key Triple DES and AES Algorithms, Section E.3. The evaluators ensure that the 10,000th value produced matches the expected value.

Implementations Conforming to NIST Special Publication 800-90A

Test 1: The evaluator shall perform 15 trials for the RNG implementation. If the RNG is configurable, the evaluator shall perform 15 trials for each configuration. The evaluator shall also confirm that the 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 shall generate eight input values for each trial. The first is a count (0 â€“ 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 shall generate eight input values for each trial. The first is a count (0 â€“ 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 paragraphs contain more information on some of the input values to be generated/selected by the evaluator.

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

Nonce: If a nonce is supported (CTR_DRBG with no Derivation Function does not use a nonce), the nonce bit length is one-half the seed length. Personalization string: The length of the personalization string must be less then 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 shall use personalization strings of two different lengths. If the implementation does not use a personalization string, no value needs to be supplied.

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

The TOE TNMS Server includes the CAVP-certified BC-FJA (Bouncy Castle FIPS Java API). See Section 1.2 for a listing of CAVP certificates.

2.1.11.2 ASPP14:FCS_RBG_EXT.2.2

TSS Assurance Activities: Documentation shall be produced - and the evaluator shall perform the activities - in accordance with Appendix D - Entropy Documentation and Assessment and the Clarification to the Entropy Documentation and Assessment Annex.

Entropy documentation has been provided to NIAP. Additionally, section 6.1 of the ST summarizes the DRGB functionality.

Guidance Assurance Activities: None Defined

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

No further testing activities are currently required.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.1.12 SSH Protocol (SSH10:FCS_SSH_EXT.1)

2.1.12.1 SSH10:FCS_SSH_EXT.1.1

TSS Assurance Activities: 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

Under SSH10:FCS_SSH_EXT.1, the TOE claims to implement SSH acting as a client with compliance to RFC 4251, 4252, 4253, 4254, 4344 (for AES-128/256-CTR), 5656 (for elliptic curve cryptography), and 6668 (for HMAC-SHA-2). As a result, SSH10:FCS_SSHC_EXT.1 is also included as per the application note. The TOE also claims conformance to the aes128-ctr (RFC 4344), aes256-ctr (RFC 4344), aes128-cbc (RFC 4253), aes256-cbc (RFC 4253) encryption algorithms and as a result those RFC conformance claims are included. The TOE claims use of hmac-sha2-256 (RFC 6668) as a data integrity MAC algorithm and that RFC claim in included. The TOE claims used of the ecdh-sha2-nistp256 (RFC 5656), ecdh-sha2-nistp384 (RFC 5656), and ecdh-sha2-nistp521 (RFC 5656) public key algorithms and those RFC claims are included. Lastly, the TOE claims to use an SSH KDF as defined in RFC 5656 (Section 5) to derive session keys and this claim is included. The TOE claims no additional algorithms or RFC conformance claims.

Guidance Assurance Activities: There are no guidance evaluation activities for this component. This SFR is evaluated by activities for other SFRs.

There are no activities for this component.

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

There are no activities for this component.

2.1.12.2 SSH10:FCS_SSH_EXT.1.2

TSS Assurance Activities: 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.

The TOE claims to support password-based and public key-based (ssh-rsa, ecdsa-sha2-nistp256, ecdsa-sha2-nistp384) authentication. This is identical to the TSS claims as Section 6.1 of the ST states the TOE supports 'password' and 'publickey' authentication methods. For public key, the TOE supports both RSA (ssh_rsa) and ECDSA (ecdsa-sha2-nistp256 and ecdsa-sha2-nistp384) methods.

The TOE claims password-based authentication methods, and as a result Section 6.1 of the ST states: the TOE uses the password option when establishing connections with managed elements supporting or requiring password authentication. The TOE uses the administrator specified password when establishing an SSH connection.

Keyboard-interactive is not a claimed authentication method included in the ST.

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

Section 4.16 DCN Management of the AGD contains information about configuring SSH which is done on a per-network element basis. Included in this section is information on configuring either "PASSWORD" or "SSH_KEY" authentication. Under the subsection *Section 4.16.3 Adding Network Elements to* TNMS, if PASSWORD is selected, this section identifies the TL1 username and password is used for SSH authentication. If SSH KEY is selected, this section identifies that a PEM format private key and encryption passphrase may be supplied.

Testing Assurance Activities: 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.

Test 1 – Not applicable, the TOE does not act as a server.

Test 2 – The TOE does act as an SSH Client. For password-based authentication, the evaluator connected to a network element via SSH using a correct password to show that the TOE was able to use this method. The evaluator attempted the same connection with an incorrect password and the connection was rejected.

For public-key authentication, the TOE supports ssh-rsa, ecdsa-sha2-nisp384, and ecdsa-sha2-nistp521. The evaluator started by configuring each with the correct credential to show that the connection could be established. The evaluator then changed the public key stored on the TOE to use an incorrect, however supported public-key algorithm than what the server was using. The resulting connection for each was rejected.

Test 3 – This test was performed in conjunction with Test 2. Under this test, the evaluator performed the following:

The evaluator then tried to configure the TOE with an unsupported key type (ssh-ed25519) and found that the TOE would refuse to initiate the connection. Similarly, the evaluator found that the TOE could not connect to a SSH server configured to use ssh-ed25519 when configured with a public key that was supported by the TOE. The evaluator attempted to connect the TOE to a SSH server using ssh-rsa-cert-v01@openssh.com, ecdsa-sha2-nistp256-cert-v01@openssh.com, ecdsa-sha2-nistp384-cert-v01@openssh.com, ecdsa-sha2-nistp521-cert-v01@openssh.com, ssh-ed25519-cert-v01@openssh.com, ssh-dss, and ssh-dss-certv01@openssh.com and found that all of these connection attempts were rejected.

2.1.12.3 SSH10:FCS_SSH_EXT.1.3

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

Section 6.1 of the ST states that the TOE's Jsch implementation tracks the size of incoming SSHv2 packets (at the transport layer) and aborts processing and discards any packet exceeding a size of 262,122 bytes.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: 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 one byte larger than the maximum size specified in this component and send it through the established SSH connection to the TOE.

c. Evaluator shall verify that the packet was dropped by the TOE by reviewing the TOE audit log for a dropped packet audit

Test 1 – The evaluator attempted to connect the TOE to a SSH server that was hardcoded to immediately send an SSH packet of 262126 bytes. The resulting connection was accepted and the SSH connection continued without error.

Test 2 – This test was performed in conjunction with Test 1. Under that test, the evaluator attempted to connect the TOE to a SSH server that was hardcoded to immediately send an SSH packet of 262127 bytes. After receiving the packet, the TOE immediately disconnected. The evaluator observed a failed message under the network element.

2.1.12.4 SSH10:FCS_SSH_EXT.1.4

TSS Assurance Activities: 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.

Section 6.1 of the ST states the TOE supports both 128 and 256 AES keys and both the CBC and CTR modes of feedback. This is identical to the claims made under SSH10:FCS_SSH_EXT.1.4

Guidance Assurance Activities: 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.

For SSH, *Section 4.16 DCN Management* of the AGD contains information about configuring SSH which is done on a per-network element basis. Included in this section are settings related to enable SSH (use secure connection), configure authentication type (PASSWORD or SSH_KEY), configuring PEM-based SSH private keys, and configuring SSH Host keys. *Section 4.16.2 SSH Algorithm Configuration* states "SSH algorithms in TNMS Client are fixed and non-configurable." Lastly, *Section 4.16.3 Adding Network Elements to TNMS* spells out all of the supported SSH key algorithms.

Additionally, *Section 4.4 Enabling FIPS mode* of the AGD indicates that the TOE installation requires setting up the cryptographic library into FIPS mode in order to be in compliance with the Common Criteria evaluated configuration. Additionally, this section also states that "the use of any other cryptographic engines, or configurations other than what is described in this section was not evaluated nor tested during the Common Criteria Evaluation of TNMS Server.

Testing Assurance Activities: 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.

Test 1 – The evaluator attempted a connection with a remote SSH server alternating with all of the encryption algorithms supported by the server. Only AES 128/256 – CTR/CBC, the claimed algorithms, were able to successful connect. The evaluator further analyzed the successful connection with aes128-cbc and verified that only the algorithms claimed in the ST for the TOE were included.

Test 2 – This was tested in conjunction in the previous test, Test 1. For each of the supported ciphers, the evaluator terminated the SSH connection via the TOE's UI and observed the TOE terminated the connection.

Test 3 -This was tested in conjunction with the previous test, Test 1. In this test, the evaluator attempted a connection with aes128-gcm@openssh.com, aes192-cbc, aes192-ctr, 3des-cbc, arcfour, arcfour128, arcfour256, blowfish-cbc, cast128-cbc, and chacha20-poly1305@openssh.com and found that each one that was not claimed by the TOE resulted in an unsuccessful connection

2.1.12.5 SSH10:FCS_SSH_EXT.1.5

TSS Assurance Activities: 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.

Section 6.1 of the ST states the TOE supports using HMAC-SHA2-256 to protect SSH packet integrity. This is identical to the claims under SSH10:FCS_SSH_EXT.1.5 as hmac-sha2-256 is the only integrity hashing algorithm claimed.

Guidance Assurance Activities: 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.

For SSH, *Section 4.16 DCN Management* of the AGD contains information about configuring SSH which is done on a per-network element basis. Included in this section are settings related to enable SSH (use secure connection), configure authentication type (PASSWORD or SSH_KEY), configuring PEM-based SSH private keys, and configuring SSH Host keys. *Section 4.16.2 SSH Algorithm Configuration* states "SSH algorithms in TNMS Client are fixed and non-configurable." Lastly, *Section 4.16.3 Adding Network Elements to TNMS* spells out all of the supported SSH key algorithms.

Additionally, *Section 4.4 Enabling FIPS mode* of the AGD indicates that the TOE installation requires setting up the cryptographic library into FIPS mode in order to be in compliance with the Common Criteria evaluated configuration. Additionally, this section also states that "the use of any other cryptographic engines, or configurations other than what is described in this section was not evaluated nor tested during the Common Criteria Evaluation of TNMS Server.

Testing Assurance Activities: 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.

Test 1 – The evaluator retested the connection with a remote SSH server, alternating with all of the message integrity hashing algorithms supported by the server. Only hmac-sha2-256 was able to successfully connect. The evaluator further analyzed the packet capture from the failed connection with hmac-sha1 and found the TOE only advertised hmac-sha2-256

Test 2 – This was tested in conjunction with the previous test, Test 1 where the evaluator tested the TOE's ability to connect with hmac-sha1, hmac-sha2-256, and hmac-sha1-96 and found they were unsuccessful. The evaluator further tested with the 'none' and hmac-md5 MAC algorithms here and found that these were also rejected.

2.1.12.6 SSH10:FCS_SSH_EXT.1.6

TSS Assurance Activities: 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.

Section 6.1 of the ST states the TOE supports ECP groups across NIST P-256/384/521 curves (i.e., ecdh-sha2-nistp256, ecdh-sha2-nistp384, ecdh-sha2-nistp521) for shared secret establishment. This is identical to the claims under SSH10:FCS_SSH_EXT.1.6.

Guidance Assurance Activities: 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.

For SSH, *Section 4.16 DCN Management* of the AGD contains information about configuring SSH which is done on a per-network element basis. Included in this section are settings related to enable SSH (use secure connection), configure authentication type (PASSWORD or SSH_KEY), configuring PEM-based SSH private keys, and configuring SSH Host keys. *Section 4.16.2 SSH Algorithm Configuration* states "SSH algorithms in TNMS Client are fixed and non-configurable." Lastly, *Section 4.16.3 Adding Network Elements to TNMS* spells out all of the supported SSH key algorithms.

Additionally, *Section 4.4 Enabling FIPS mode* of the AGD indicates that the TOE installation requires setting up the cryptographic library into FIPS mode in order to be in compliance with the Common Criteria evaluated configuration. Additionally, this section also states that "the use of any other cryptographic engines, or configurations other than what is described in this section was not evaluated nor tested during the Common Criteria Evaluation of TNMS Server.

Testing Assurance Activities: 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.

Test 1 – The evaluator retested the connection with a remote SSH server, alternating with all of the key exchange algorithms supported by the server. Only ecdh-sha2-nistp256, ecdh-sha2-nistp384, and ecdh-sha2-nistp521 were able to successfully connect. The evaluator further analyzed the packet capture from the failed connection with dh-group-14-sha1 and found that only the claimed algorithms were advertised.

Test 2 – This was tested in conjunction with the previous test, Test 1 where the evaluator tested the TOE's ability to connect using diffie-hellman-group14-sha1, curve25519-sha256@libssh.org, diffie-hellman-group1-sha1, diffie-hellman-group-exchange-sha1, and diffie-hellman-group-exchange-sha256 and found that they were unsuccessful.

2.1.12.7 SSH10:FCS_SSH_EXT.1.7

TSS Assurance Activities: 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

Section 6.1 of the ST states the TOE supports the KDF defined RFC 5656 for ECDH. This is identical to the claims made under SSH10:FCS_SSH_EXT.1.7

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.12.8 SSH10:FCS_SSH_EXT.1.8

TSS Assurance Activities: 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.

Section 6.1 of the ST states the TOE enforces a rekeying of the SSH session at the thresholds defined by the SFR which the SFR provides the non-optional limits of one hour connection time and no more than one gigabyte of transmitted/received data. The TOE is a software application designed to run generically on multiple platforms and does not feature any hardware limitations.

Guidance Assurance Activities: 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.

For SSH, *Section 4.16 DCN Management* of the AGD contains information about configuring SSH which is done on a per-network element basis. *Section 4.16.2 SSH Algorithm Configuration* states "SSH algorithms in TNMS Client are fixed and non-configurable. Similarly, SSH rekeying is always enabled and not configurable."

Additionally, *Section 4.4 Enabling FIPS mode* of the AGD indicates that the TOE installation requires setting up the cryptographic library into FIPS mode in order to be in compliance with the Common Criteria evaluated configuration. Additionally, this section also states that "the use of any other cryptographic engines, or configurations other than what is described in this section was not evaluated nor tested during the Common Criteria Evaluation of TNMS Server.

Testing Assurance Activities: 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.

Test 1 – The evaluator initiated a SSH connection from the TOE to an external SSH server and maintained the connection for over an hour. The evaluator observed log messages on the TOE that the session initiated around 19:01:19 and a rekey was initiated from the TOE approximately 40 minutes later at 19:40:37. This timeout for rekeying was confirmed by the vendor and is under the required one hour

Test 2 – The TOE does not have a hardware-based limitation for generating traffic, however it did have a software-based limitation for generating enough traffic to cause a rekey before the time-based rekey mechanism. The vendor exposed the configuration file where the rekey limit was set for purposes of testing. The evaluator configured the SSH connection transmit limit to 5KB and 10KB separately, established a connection to a network element, and found that the TOE initiated a rekey once the threshold was exceeded. As a result, the threshold limit is enforced and the default value of 750KB would be enforced prior to generating 1GB of traffic.

Test 3 - The TOE has the same limitations as Test 2. The evaluator configured the SSH connection transmit limit to 5KB and 131KB separately, established a connection to a network element, and found that the TOE initiated a rekey once the threshold was exceeded. As a result, the threshold limit is enforced and the default value of 750KB would be enforced prior to generating 1GB of traffic.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.1.13 SSH Protocol - Client (SSH10:FCS_SSHC_EXT.1)

2.1.13.1 SSH10:FCS_SSHC_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: 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.

For SSH, *Section 4.16 DCN Management* of the AGD contains information about configuring SSH which is done on a per-network element basis. Included in this section are settings related to enable SSH (use secure connection), configure authentication type (PASSWORD or SSH_KEY), configuring PEM-based SSH private keys, and configuring SSH Host keys. *Section 4.16.2 SSH Algorithm Configuration* states "SSH algorithms in TNMS Client are fixed and non-configurable." Lastly, *Section 4.16.3 Adding Network Elements to TNMS* spells out all of the supported SSH key algorithms.

Additionally, *Section 4.4 Enabling FIPS mode* of the AGD indicates that the TOE installation requires setting up the cryptographic library into FIPS mode in order to be in compliance with the Common Criteria evaluated configuration. Additionally, this section also states that "the use of any other cryptographic engines, or configurations other than what is described in this section was not evaluated nor tested during the Common Criteria Evaluation of TNMS Server.

Component Testing Assurance Activities: 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

Test 1 – The evaluator configured the TOE with a single host name and public key that matched an external SSH server. The evaluator initiated a connection from the TOE to the matching server and found that the connection was successful.

Test 2 – The evaluator modified a byte in the previous host key so that it no longer matched the external SSH server. The evaluator initiated a connection from the TOE to the now non-matching server and found the connection was unsuccessful.

Test 3 – The evaluator removed all entries in the SSH host key store and initiated a connection from the TOE to the external server. The evaluator observed that the connection was unsuccessful, however the TOE did present the option later to adopt the last used host key into the local database and then the user could initiate a new connection.

Test 4 – Not applicable, the TOE does not use a list of trusted certification authorities

Test 5 – Not applicable, the TOE does not use a list of trusted certification authorities

2.1.14 Storage of Credentials (ASPP14:FCS_STO_EXT.1)

2.1.14.1 ASPP14:FCS_STO_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall check the TSS to ensure that it lists all persistent credentials (secret keys, PKI private keys, or passwords) needed to meet the requirements in the ST. For each of these items, the evaluator shall confirm that the TSS lists for what purpose it is used, and how it is stored.

Section 6.1 contains the following pertaining to persistent credentials: The TNMS server stores administrative passwords, TNMS's TLS server certificates (and private keys), and SSHv2 private keys and passwords. The administrative password is used to access the TOE from the external TNMS Client. The TLS certificates are used to authenticate the TLS channel the TOE and the TNMS client use to communicate. The SSH authentication keys are used to authenticate the TOE to any configured network element.

In terms of how these are stored, the same section identifies that the TOE does not store administrative passwords directly, but instead, derives a value from the received from the TNMS client using PBKDF, and stores that derived value in its configuration database. The TOE compares these stored values to those newly derived from the administrator's password received from the TNMS Client to decide whether to grant access to the Client. The TOE

protects private keys (TLS server and SSH client) by storing them in its Bouncy Castle keystore in the OS provided filesystem.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: For all credentials for which the application implements functionality, the evaluator shall verify credentials are encrypted according to FCS_COP.1/SKC or conditioned according to FCS_CKM.1.1/AK and FCS_CKM.1/PBKDF. For all credentials for which the application invokes platform-provided functionality, the evaluator shall perform the following actions which vary per platform.

Platforms: Android....

The evaluator shall verify that the application uses the Android KeyStore or the Android KeyChain to store certificates.

Platforms: Microsoft Windows....

The evaluator shall verify that all certificates are stored in the Windows Certificate Store. The evaluator shall verify that other credentials, like passwords, are stored in the Windows Credential Manager or stored using the Data Protection API (DPAPI). For Windows Universal Applications, the evaluator shall verify that the application is using the ProtectData class and storing credentials in IsolatedStorage.

Platforms: Apple iOS....

The evaluator shall verify that all credentials are stored within a Keychain.

Platforms: Linux....

The evaluator shall verify that all keys are stored using Linux keyrings.

Platforms: Oracle Solaris....

The evaluator shall verify that all keys are stored using Solaris Key Management Framework (KMF).

Platforms: Apple macOS....

The evaluator shall verify that all credentials are stored within Keychain

Test 1 – The evaluator tested the administrative password, server certificate/private key, and SSHv2 private keys/ passwords separately. All credentials implement functionality to be securely stored according to FCS_COP.1/SKC or FCS_CKM.1/PBDKF. The evaluator configured a known administrative password and then searched through all of the directories known to interact with the TOE application. The evaluator found no evidence of the plaintext password in memory. TLS private keys are stored in a java keystore file. The evaluator performed a hexdump on the keystore and searched for the matching private key, however the keystore was encrypted and the private key could not be found. The evaluator attempted to use the keytool list command used to view the keystore, however without the password needed to decrypt it, and found that the key could not be accessed. The evaluator configured a SSH Client private key for a network element. The evaluator searched the directories the TOE could

write to and found no evidence of the key in memory. The evaluator then repeated this process by configuring a SSH password for a network element and searching the same directories. The evaluator found no evidence of the values in memory and concluded that they must be stored in an encrypted form.

2.1.15 TLS Protocol (PKGTLS11:FCS_TLS_EXT.1)

2.1.15.1 PKGTLS11:FCS_TLS_EXT.1.1

TSS Assurance Activities: None Defined

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

The evaluator examined both the ST and the AGD and only found claims to the TOE operating as a TLS Server. As a result, PKGTLS11:FCS_TLSS_EXT.1 is claimed in the ST. The evaluator found no references in the ST or AGD to indicate support for mutual authentication (PKGTLS11:FCS_TLSS_EXT.2), limiting of hashing algorithms by the server (PKGTLS11:FCS_TLSS_EXT.3), renegotiation (PKGTLS11:FCS_TLSS_EXT.4), TLS as a client (PKGTLS11:FCS_TLSC_EXT.*), or DTLS (PKGTLS11:FCS_DTLS*). The evaluator further examined the TSS and AGD and found no claims that contradicted any of the more specific claims under this SFR.

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.1.16 TLS Server Protocol (PKGTLS11:FCS_TLSS_EXT.1)

2.1.16.1 PKGTLS11:FCS_TLSS_EXT.1.1

TSS Assurance Activities: 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.

Section 6.1 of the ST contains the following regarding TLS ciphersuites which match the selection in the ST: The TOE supports the ciphersuites listed in section **Error! Reference source not found.** (ECDHE with AES GCM cipher suites) and the TOE selects the elliptic curve (NIST P-256, 834, or 521) depending upon the supported curves specified in the client hello message.

Guidance Assurance Activities: The evaluator shall also check the operational guidance to ensure that it contains instructions on configuring the TOE so that TLS conforms to the description in the TSS

For TLS, *Section 4.5 Managing Certificates and Keys* of the AGD contains information about configuring TLS server certificates. Additionally, *Section 4.5.3 TLS Configuration* also states "Other than setting the trusted root certificates, TLS settings in TNMS Server – such as supported cipher suites and key sizes - are fixed and not configurable by the user."

Additionally, *Section 4.4 Enabling FIPS mode* of the AGD indicates that the TOE installation requires setting up the cryptographic library into FIPS mode in order to be in compliance with the Common Criteria evaluated configuration. Additionally, this section also states that "the use of any other cryptographic engines, or configurations other than what is described in this section was not evaluated nor tested during the Common Criteria Evaluation of TNMS Server.

Testing Assurance Activities: 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 evaluator shall send a Client Hello to the server with a list of cipher suites that does not contain any of the cipher suites in the server's ST and verify that the server denies the connection. Additionally, the evaluator shall send a Client Hello to the server containing only the TLS_NULL_WITH_NULL_NULL cipher suite and verify that the server denies the connection.

Test 3: If RSA key exchange is used in one of the selected ciphersuites, the evaluator shall use a client to send a properly constructed Key Exchange message with a modified EncryptedPreMasterSecret field during the TLS handshake. The evaluator shall verify that the handshake is not completed successfully and no application data flows.

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

Test 4.1: Removed per TD0469

Test 4.2: Modify a byte in the data of the client's Finished handshake message, and verify that the server rejects the connection and does not send any application data.

Test 4.3: Demonstrate that the TOE will not resume a session for which the client failed to complete the handshake (independent of TOE support for session resumption):

Test 4.3i [conditional]: If the TOE does not support session resumption based on session IDs according to RFC4346 (TLS1.1) or RFC5246 (TLS1.2) or session tickets according to RFC5077, the evaluator shall perform the following test:

a) The evaluator shall send a Client Hello with a zero-length session identifier and with a SessionTicket extension containing a zero-length ticket.

b) The evaluator shall verify the server does not send a NewSessionTicket handshake message (at any point in the handshake).

c) The evaluator shall verify the Server Hello message contains a zero-length session identifier or passes the following steps:

Note: The following steps are only performed if the ServerHello message contains a non-zero length SessionID.

d) The evaluator shall complete the TLS handshake and capture the SessionID from the ServerHello.

e) The evaluator shall send a ClientHello containing the SessionID captured in step d). This can be done by keeping the TLS session in step d) open or start a new TLS session using the SessionID captured in step d).

f) The evaluator shall verify the TOE (1) implicitly rejects the SessionID by sending a ServerHello containing a different SessionID and by performing a full handshake (as shown in Figure 1 of RFC 4346 or RFC 5246), or (2) terminates the connection in some way that prevents the flow of application data.

Test 4.3ii [conditional]: If the TOE supports session resumption using session IDs according to RFC4346 (TLS1.1) or RFC5246 (TLS1.2), the evaluator shall carry out the following steps (note that for each of these tests, it is not necessary to perform the test case for each supported version of TLS):

a) The evaluator shall conduct a successful handshake and capture the TOE-generated session ID in the Server Hello message. The evaluator shall then initiate a new TLS connection and send the previously captured session ID to show that the TOE resumed the previous session by responding with ServerHello containing the same SessionID immediately followed by ChangeCipherSpec and Finished messages (as shown in Figure 2 of RFC 4346 or RFC 5246).

b) The evaluator shall initiate a handshake and capture the TOE-generated session ID in the Server Hello message. The evaluator shall then, within the same handshake, generate or force an unencrypted fatal Alert message immediately before the client would otherwise send its ChangeCipherSpec message thereby disrupting the handshake. The evaluator shall then initiate a new Client Hello using the previously captured session ID, and verify that the server (1) implicitly rejects the session ID by sending a ServerHello containing a different SessionID and performing a full handshake (as shown in figure 1 of RFC 4346 or RFC 5246), or (2) terminates the connection in some way that prevents the flow of application data.

Test 4.3iii [conditional]: If the TOE supports session tickets according to RFC5077, the evaluator shall carry out the following steps (note that for each of these tests, it is not necessary to perform the test case for each supported version of TLS):

a) The evaluator shall permit a successful TLS handshake to occur in which a session ticket is exchanged with the non-TOE client. The evaluator shall then attempt to correctly reuse the previous session by sending the session ticket in the ClientHello. The evaluator shall confirm that the TOE responds with a ServerHello with an empty SessionTicket extension, NewSessionTicket, ChangeCipherSpec and Finished messages (as seen in figure 2 of RFC 5077).

b) The evaluator shall permit a successful TLS handshake to occur in which a session ticket is exchanged with the non-TOE client. The evaluator will then modify the session ticket and send it as part of a new Client Hello message. The evaluator shall confirm that the TOE either (1) implicitly rejects the session ticket by performing a full handshake (as shown in figure 3 or 4 of RFC 5077), or (2) terminates the connection in some way that prevents the flow of application data.

Test 4.4: Send a message consisting of random bytes from the client after the client has issued the ChangeCipherSpec message and verify that the server denies the connection.

(TD0588 applied)

Test 1 – The evaluator made a TLS connection using each of the claimed ciphersuites. The evaluator was able to capture each ciphersuite using a packet capture

Test 2 – The evaluator configured a client to use TLS. The evaluator then attempted to require a list of ciphersuites not supported by the TOE and the attempt was rejected. The evaluator attempted to require the TLS_NULL_WITH_NULL_NULL ciphersuite and the attempt was rejected.

Test 3 – Not applicable, this test only applies if RSA key exchange is used and the TOE only uses ECDHE

Test 4 – The evaluator configured a client to use TLS. The evaluator then created packets that modified the Client's Finished Handshake message. The connection was denied as expected. The TOE does not support session resumption based on session IDs or session tickets so test 4.3i applies. The evaluator sent the TOE a zero length session ID in the server hello and observed the server did not send a NewSessionTicket or a non-zero-length sessionID. The evaluator then created packets that sent random byes in place of a Finished message and observed the connection was rejected.

2.1.16.2 PKGTLS11:FCS_TLSS_EXT.1.2

TSS Assurance Activities: The evaluator shall verify that the TSS contains a description of the denial of old SSL and TLS versions consistent relative to selections in FCS_TLSS_EXT.1.2.

Section 6.1 of the ST identifies that the TOE supports only TLS version 1.2 and rejects connection attempts from all prior TLS versions.

Guidance Assurance Activities: The evaluator shall verify that the AGD guidance includes any configuration necessary to meet this requirement.

For TLS, *Section 4.5 Managing Certificates and Keys* of the AGD contains information about configuring TLS server certificates. Additionally, *Section 4.5.3 TLS Configuration* also states "Other than setting the trusted root certificates, TLS settings in TNMS Server – such as supported cipher suites and key sizes - are fixed and not configurable by the user."

Additionally, *Section 4.4 Enabling FIPS mode* of the AGD indicates that the TOE installation requires setting up the cryptographic library into FIPS mode in order to be in compliance with the Common Criteria evaluated configuration. Additionally, this section also states that "the use of any other cryptographic engines, or configurations other than what is described in this section was not evaluated nor tested during the Common Criteria Evaluation of TNMS Server.

Testing Assurance Activities: Test 1: The evaluator shall send a Client Hello requesting a connection with version SSL 2.0 and verify that the server denies the connection. The evaluator shall repeat this test with SSL 3.0 and TLS 1.0, and TLS 1.1 if it is selected.

The evaluator configured a client to use TLS. The evaluator then attempted to initiate the client's connection using SSL 2.0, SSL 3.0, TLS 1.0, TLS 1.1, and TLS 1.2. The TOE accepted the client connection with TLS 1.2 and rejected every other connection attempt.

2.1.16.3 PKGTLS11:FCS_TLSS_EXT.1.3

TSS Assurance Activities: The evaluator shall verify that the TSS describes the key agreement parameters of the server's Key Exchange message

The TOE only claims support for ECDHE using secp256r1, secp384r1, or secp521r1. Section 6.1 of the ST states the TOE supports the ciphersuites listed in section **Error! Reference source not found.** (ECDHE with AES GCM cipher suites) and the TOE selects the elliptic curve (NIST P-256, 834, or 521) depending upon the supported curves specified in the client hello message.

Guidance Assurance Activities: The evaluator shall verify that any configuration guidance necessary to meet the requirement must be contained in the AGD guidance.

For TLS, *Section 4.5 Managing Certificates and Keys* of the AGD contains information about configuring TLS server certificates. Additionally, *Section 4.5.3 TLS Configuration* also states "Other than setting the trusted root certificates, TLS settings in TNMS Server – such as supported cipher suites and key sizes - are fixed and not configurable by the user."

Additionally, *Section 4.4 Enabling FIPS mode* of the AGD indicates that the TOE installation requires setting up the cryptographic library into FIPS mode in order to be in compliance with the Common Criteria evaluated configuration. Additionally, this section also states that "the use of any other cryptographic engines, or configurations other than what is described in this section was not evaluated nor tested during the Common Criteria Evaluation of TNMS Server.

Testing Assurance Activities: The evaluator shall conduct the following tests. The testing can be carried out manually with a packet analyzer or with an automated framework that similarly captures such empirical evidence. Note that this testing can be accomplished in conjunction with other testing activities. For each of the following tests, determining that the size matches the expected size is sufficient.

Test 1: [conditional] If RSA-based key establishment is selected, the evaluator shall attempt a connection using RSA-based key establishment with a supported size. The evaluator shall verify that the size used matches that which is configured. The evaluator shall repeat this test for each supported size of RSA-based key establishment.

Test 2: [conditional] If finite-field (i.e. non-EC) Diffie-Hellman ciphers are selected, the evaluator shall attempt a connection using a Diffie-Hellman key exchange with a supported parameter size or supported group. The evaluator shall verify that the key agreement parameters in the Key Exchange message are the ones configured. The evaluator shall repeat this test for each supported parameter size or group.

Test 3: [conditional] If ECDHE ciphers are selected, the evaluator shall attempt a connection using an ECDHE ciphersuite with a supported curve. The evaluator shall verify that the key agreement parameters in the Key Exchange message are the ones configured. The evaluator shall repeat this test for each supported elliptic curve.

Test 1 – Not applicable, the TOE only supports ECDHE for TLS server connections

Test 2 – Not applicable, the TOE only supports ECDHE for TLS server connections

Test 3 – The evaluator made a TLS connection using each of the claimed curves. The evaluator was able to capture each key size or curve using a packet capture.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2 User data protection (FDP)

2.2.1 Encryption Of Sensitive Application Data (ASPP14:FDP_DAR_EXT.1)

2.2.1.1 ASPP14:FDP_DAR_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to ensure that it describes the sensitive data processed by the application. The evaluator shall then ensure that the following activities cover all of the sensitive data identified in the TSS. If not store any sensitive data is selected, the evaluator shall inspect the TSS to ensure that it describes how sensitive data cannot be written to non-volatile memory. The evaluator shall also ensure that this is consistent with the filesystem test below.

Section 6.2 of the TSS states the TOE does not process any sensitive data outside of the credentials claimed as a part of FCS_STO_EXT.1. The TOE stores sensitive data including the server login password and SSH client credentials within its configuration database. The TOE also has a TLS Server Certificate and private key used for incoming TLS connections stored in a Bouncy Castle-encrypted Keystore stored in the TOE's installation directory.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: Evaluation activities (after the identification of the sensitive data) are to be performed on all sensitive data listed that are not covered by FCS_STO_EXT.1. The evaluator shall inventory the filesystem locations where the application may write data. The evaluator shall run the application and attempt to store sensitive data. The evaluator shall then inspect those areas of the filesystem to note where data was stored (if any), and determine whether it has been encrypted.

If 'leverage platform-provided functionality' is selected, the evaluation activities will be performed as stated in the following requirements, which vary on a per-platform basis.

Platforms: Android....

The evaluator shall inspect the TSS and verify that it describes how files containing sensitive data are stored with the MODE_PRIVATE flag set.

Platforms: Microsoft Windows....

The Windows platform currently does not provide data-at-rest encryption services which depend upon invocation by application developers. The evaluator shall verify that the Operational User Guidance makes the need to activate platform encryption, such as BitLocker or Encrypting File System (EFS), clear to the end user.

Platforms: Apple iOS....

The evaluator shall inspect the TSS and ensure that it describes how the application uses the Complete Protection, Protected Unless Open, or Protected Until First User Authentication Data Protection Class for each data file stored locally.

Platforms: Linux....

The Linux platform currently does not provide data-at-rest encryption services which depend upon invocation by application developers. The evaluator shall verify that the Operational User Guidance makes the need to activate platform encryption clear to the end user.

Platforms: Oracle Solaris....

The Solaris platform currently does not provide data-at-rest encryption services which depend upon invocation by application developers. The evaluator shall verify that the Operational User Guidance makes the need to activate platform encryption clear to the end user.

Platforms: Apple macOS....

The macOS platform currently does not provide data-at-rest encryption services which depend upon invocation by application developers. The evaluator shall verify that the Operational User Guidance makes the need to activate platform encryption clear to the end user.

The evaluator noted that the TOE does not store any sensitive data outside of data covered by FCS_STO_EXT.1. Since there is no sensitive data that this evaluation activity refers to, this test is not applicable and otherwise the sensitive data covered under FCS_STO_EXT.1 is tested under FCS_STO_EXT.1-t1.

2.2.2 Access to Platform Resources (ASPP14:FDP_DEC_EXT.1)

2.2.2.1 ASPP14:FDP_DEC_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: The evaluator shall perform the platform-specific actions below and inspect user documentation to determine the application's access to hardware resources. The evaluator shall ensure that this is consistent with the selections indicated. The evaluator shall review documentation provided by the application developer and for each resource which it accesses, identify the justification as to why access is required.

Section 2.2 Other Hardware and Software resources of the AGD indicates that the TNMS Server does not require access to any sensitive information repositories or special hardware resources other than network connectivity used to communicate with the TNMS Client, configured network elements, or check for updates.

Testing Assurance Activities: Platforms: Android....

The evaluator shall verify that each uses-permission entry in the AndroidManifest.xml file for access to a hardware resource is reflected in the selection.

Platforms: Microsoft Windows....

For Windows Universal Applications the evaluator shall check the WMAppManifest.xml file for a list of required hardware capabilities. The evaluator shall verify that the user is made aware of the required hardware capabilities when the application is first installed. This includes permissions such as ID_CAP_ISV_CAMERA, ID_CAP_LOCATION, ID_CAP_NETWORKING, ID_CAP_MICROPHONE, ID_CAP_PROXIMITY and so on. A complete list of Windows App permissions can be found at:

http://msdn.microsoft.com/en-US/library/windows/apps/jj206936.aspx

For Windows Desktop Applications the evaluator shall identify in either the application software or its documentation the list of the required hardware resources.

Platforms: Apple iOS....

The evaluator shall verify that either the application or the documentation provides a list of the hardware resources it accesses.

Platforms: Linux....

The evaluator shall verify that either the application software or its documentation provides a list of the hardware resources it accesses.

Platforms: Oracle Solaris....

The evaluator shall verify that either the application software or its documentation provides a list of the hardware resources it accesses.

Platforms: Apple macOS....

The evaluator shall verify that either the application software or its documentation provides a list of the hardware resources it accesses.

The TOE only claims access to network connectivity and does not make any claims for access to additional hardware resources. The evaluator never saw any special requests for hardware resources during any of the installation or operational screens for the TOE.

2.2.2.2 ASPP14:FDP_DEC_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: The evaluator shall perform the platform-specific actions below and inspect user documentation to determine the application's access to sensitive information repositories. The evaluator shall ensure that this is consistent with the selections indicated. The evaluator shall review documentation provided by the application developer and for each sensitive information repository which it accesses, identify the justification as to why access is required.

Section 2.2 Other Hardware and Software resources of the AGD indicates that the TNMS Server does not require access to any sensitive information repositories or special hardware resources other than network connectivity used to communicate with the TNMS Client, configured network elements, or check for updates.

Testing Assurance Activities: Platforms: Android....

The evaluator shall verify that each uses-permission entry in the AndroidManifest.xml file for access to a sensitive information repository is reflected in the selection.

Platforms: Microsoft Windows....

For Windows Universal Applications the evaluator shall check the WMAppManifest.xml file for a list of required capabilities. The evaluator shall identify the required information repositories when the application is first installed. This includes permissions such as ID_CAP_CONTACTS,ID_CAP_APPOINTMENTS,ID_CAP_MEDIALIB and so on. A complete list of Windows App permissions can be found at:

http://msdn.microsoft.com/en-US/library/windows/apps/jj206936.aspx

For Windows Desktop Applications the evaluator shall identify in either the application software or its documentation the list of sensitive information repositories it accesses.

Platforms: Apple iOS....

The evaluator shall verify that either the application software or its documentation provides a list of the sensitive information repositories it accesses.

Platforms: Linux....

The evaluator shall verify that either the application software or its documentation provides a list of sensitive information repositories it accesses.

Platforms: Oracle Solaris....

The evaluator shall verify that either the application software or its documentation provides a list of sensitive information repositories it accesses.

Platforms: Apple macOS....

The evaluator shall verify that either the application software or its documentation provides a list of sensitive information repositories it accesses.

The TOE does not make any claims for access to additional sensitive information repositories. The evaluator never saw any special requests for sensitive data repositories during any of the installation or operational screens for the TOE.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.3 Network Communications (ASPP14:FDP_NET_EXT.1)

2.2.3.1 ASPP14:FDP_NET_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The evaluator shall perform the following tests:

Test 1: The evaluator shall run the application. While the application is running, the evaluator shall sniff network traffic ignoring all non-application associated traffic and verify that any network communications witnessed are documented in the TSS or are user-initiated.

Test 2: The evaluator shall run the application. After the application initializes, the evaluator shall run network port scans to verify that any ports opened by the application have been captured in the ST for the third selection and its assignment. This includes connection-based protocols (e.g. TCP, DCCP) as well as connectionless protocols (e.g. UDP).

Platforms: Android....

If 'no network communication' is selected, the evaluator shall ensure that the application's AndroidManifest.xml file does not contain a <uses-permission> or <uses-permission-sdk-23> tag containing android:name='android.permission.INTERNET'. In this case, it is not necessary to perform the above Tests 1 and 2, as the platform will not allow the application to perform any network communication.

Test 1 – The evaluator captured traffic while the TOE was used to communicate with the external TNMS Client, communicate with the external network element, and check for version. The evaluator analyzed the packet capture and found only the mentioned network traffic from the TOE process.

Test 2 – The evaluator performed a port scan and verified that the TOE does not open any port aside from the documented port 8443 which is used for responding to connection attempts from TNMS Clients.

2.3 Security management (FMT)

2.3.1 Secure by Default Configuration (ASPP14:FMT_CFG_EXT.1)

2.3.1.1 ASPP14:FMT_CFG_EXT.1.1

TSS Assurance Activities: The evaluator shall check the TSS to determine if the application requires any type of credentials and if the application installs with default credentials.

Section 6.3 of the ST states that the TOE requires that administrators (when they connected through a TNMS Client) authenticate themselves. The TOE comes with a default administrative user and password, and during the initial configuration of the TOE, the initial administrator must set a new password, after which the TOE no longer accepts the default password.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: If the application uses any default credentials the evaluator shall run the following tests.

Test 1: The evaluator shall install and run the application without generating or loading new credentials and verify that only the minimal application functionality required to set new credentials is available.

Test 2: The evaluator shall attempt to clear all credentials and verify that only the minimal application functionality required to set new credentials is available.

Test 3: The evaluator shall run the application, establish new credentials and verify that the original default credentials no longer provide access to the application.

Test 1 – The evaluator installed the TOE and attempted to log on for the first time. The only option presented to the evaluator was to change the default password. The evaluator attempted to circumvent the change password prompt, however the evaluator was not successfully logged in and could not perform any actions on the TOE other than attempt to login again and change the password

Test 2 – Not applicable, the TOE does not provide a method to clear all credentials

Test 3 – This is tested in conjunction with Test 1. The evaluator changed the default password and attempted to log in again with the default password, however the login attempt failed.

2.3.1.2 ASPP14:FMT_CFG_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator shall install and run the application. The evaluator shall inspect the filesystem of the platform (to the extent possible) for any files created by the application and ensure that their permissions are adequate to protect them. The method of doing so varies per platform.

Platforms: Android....

The evaluator shall run the command find -L . -perm /002 inside the application's data directories to ensure that all files are not world-writable. The command should not print any files.

Platforms: Microsoft Windows....

The evaluator shall run the SysInternals tools, Process Monitor and Access Check (or tools of equivalent capability, like icacls.exe) for Classic Desktop applications to verify that files written to disk during an application's installation have the correct file permissions, such that a standard user cannot modify the application or its data files. For Windows Universal Applications the evaluator shall consider the requirement met because of the AppContainer sandbox.

Platforms: Apple iOS....

The evaluator shall determine whether the application leverages the appropriate Data Protection Class for each data file stored locally.

Platforms: Linux....

The evaluator shall run the command find -L. -perm /002 inside the application's data directories to ensure that all files are not world-writable. The command should not print any files.

Platforms: Oracle Solaris....

The evaluator shall run the command find . -perm -002 inside the application's data directories to ensure that all files are not world-writable. The command should not print any files.

Platforms: Apple macOS....

The evaluator shall run the command find . -perm +002 inside the application's data directories to ensure that all files are not world-writable. The command should not print any files.

The evaluator ran the prescribed find command in the Linux directories associated with the TOE including the application's data directory. The evaluator found no files returned by the command.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.3.2 Supported Configuration Mechanism - per TD0624 (ASPP14:FMT_MEC_EXT.1)

2.3.2.1 ASPP14:FMT_MEC_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall review the TSS to identify the application's configuration options (e.g. settings) and determine whether these are stored and set using the mechanisms supported by the platform or implemented by the application in accordance with the PP-Module for File Encryption. At a minimum the TSS shall list settings related to any SFRs and any settings that are mandated in the operational guidance in response to an SFR.

Conditional: If 'implement functionality to encrypt and store configuration options as defined by FDP_PRT_EXT.1 in the PP-Module for File Encryption' is selected, the evaluator shall ensure that the TSS identifies those options, as well as indicates where the encrypted representation of these options is stored.

Section 6.3 of the ST states the TOE makes use of its application storage area within its Linux operating system to store its configuration database containing all persistent information. This encompasses all configuration settings on the TOE. The TOE does not implement any functionality to store configuration options, but instead relies on the platform vendor for storing and setting configuration options.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: If 'invoke the mechanisms recommended by the platform vendor for storing and setting configuration options' is chosen, the method of testing varies per platform as follows:

Platforms: Android....

The evaluator shall run the application and make security-related changes to its configuration. The evaluator shall check that at least one file exists at location /data/data/package/shared_prefs/ (for SharedPreferences) and/or /data/data/package/files/datastore (for DataStore), where the package is the Java package of the application. For SharedPreferences the evaluator shall examine the XML file to make sure it reflects the changes made to the configuration to verify that the application used SharedPreferences and/or PreferenceActivity to store the configuration data. For DataStore the evaluator shall use a protocol buffer analyzer to examine the file to make sure it reflects the changes made to the configuration to verify that the application used DataStore to store the configuration data.

Platforms: Microsoft Windows....

The evaluator shall determine and verify that Windows Universal Applications use either the Windows.Storage namespace, Windows.UI.ApplicationSettings namespace or the IsolatedStorageSettings namespace for storing application specific settings. For .NET applications, the evaluator shall determine and verify that the application uses one of the locations listed in https://docs.microsoft.com/en-us/dotnet/framework/configure-apps/ for storing application specific settings. For Classic Desktop applications, the evaluator shall run the application while monitoring it with the SysInternals tool ProcMon and make changes to its configuration. The evaluator shall verify that ProcMon logs show corresponding changes to the Windows Registry or C:directory.

Platforms: Apple iOS....

The evaluator shall verify that the app uses the user defaults system or key-value store for storing all settings.

Platforms: Linux....

The evaluator shall run the application while monitoring it with the utility strace. The evaluator shall make security-related changes to its configuration. The evaluator shall verify that strace logs corresponding changes to configuration files that reside in /etc (for system-specific configuration), in the user's home directory (for user-specific configuration), or /var/lib/ (for configurations controlled by UI and not intended to be directly modified by an administrator).

Platforms: Oracle Solaris....

The evaluator shall run the application while monitoring it with the utility dtrace. The evaluator shall make security-related changes to its configuration. The evaluator shall verify that dtrace logs corresponding changes to configuration files that reside in /etc (for system-specific configuration) or in the user's home directory (for user-specific configuration).

Platforms: Apple macOS....

The evaluator shall verify that the application stores and retrieves settings using the NSUserDefaults class.

If ' implement functionality to encrypt and store configuration options as defined by FDP_PRT_EXT.1 in the PP-Module for File Encryption' is selected, for all configuration options listed in the TSS as being stored and protected using encryption, the evaluator shall examine the contents of the configuration option storage (identified in the TSS) to determine that the options have been encrypted.

The evaluator ran the test for invoke platform provided mechanisms.

The evaluator ran the application and modified the SSH connection for a network element while monitoring the various process id for the application and the external oracle database program. After examining the strace output, the evaluator concluded that the TOE was storing the configuration data in the external database configured to be installed under /var/lib/oradata.

2.3.3 Specification of Management Functions (ASPP14:FMT_SMF.1)

2.3.3.1 ASPP14:FMT_SMF.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: The evaluator shall verify that every management function mandated by the PP is described in the operational guidance and that the description contains the information required to perform the management duties associated with the management function.

For Function 1 – Configure TLS Server Credentials, *Section 4.5 Managing Certificates and Keys* of the AGD contains instructions for adding to and viewing the contents of the TOE's trusted root certificate store.

For Function 2 – Configure Network Elements, Section *4.16 DCN Management* of the AGD contains instructions for adding network elements such as configuring a MVM mediator, adding a channel, and adding a network element with the ID name, TL1 ID, IP address, port, username/password, ssh key, authentication method, and host key.

Additionally, this same section contains information about configuring the SSH channel to be Common Criteria compliant such as enabling secure host checking and enabling secure connections.

Component Testing Assurance Activities: The evaluator shall test the application's ability to provide the management functions by configuring the application 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.

The evaluator tested each of the TOE's management functions to ensure that it could be configured and behaved as expected.

Function 1 – Configure TLS Sever Credentials – The evaluator added a TLS server certificate to the TOE keystore and restarted the TOE server. Using an external TNMS client, the evaluator demonstrated that the correct certificate was being used. Further testing against the TLS server certificate ensuring the correct one is being used is performed under PKGTLS11:FCS_TLSS_EXT.1.

Function 2 – Configure Network Elements – The evaluator add a network element configuration under the TOE and configured a IP address, port number, application layer TID, username, password, SSH authentication method, and SSH Host Key. Further testing against similar configurations to ensure the configuration is correct is done under SSH10:FCS_SSH_EXT.1

2.4 Privacy (FPR)

2.4.1 User Consent for Transmission of Personally Identifiable (ASPP14:FPR_ANO_EXT.1)

2.4.1.1 ASPP14:FPR_ANO_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall inspect the TSS documentation to identify functionality in the application where PII can be transmitted.

Section 6.4 of the ST states that the TOE does not transmit any PII.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: If require user approval before executing is selected, the evaluator shall run the application and exercise the functionality responsibly for transmitting PII and verify that user approval is required before transmission of the PII.

Not applicable, the TOE does not transmit PII.

2.5 Protection of the TSF (FPT)

2.5.1 Anti-Exploitation Capabilities (ASPP14:FPT_AEX_EXT.1)

2.5.1.1 ASPP14:FPT_AEX_EXT.1.1

TSS Assurance Activities: The evaluator shall ensure that the TSS describes the compiler flags used to enable ASLR when the application is compiled.

The TOE is primarily a Java application that does not rely on compilation flags for most runtime protections including ASLR.

For native executables included with the TOE, section 6.5 of the ST states the corresponding compilation flags are used including /DYNAMICBASE, /GS, and /NXCOMPAT for PE and -pie (position independent executable) and -fstack-protector-all for ELF executables.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator shall perform either a static or dynamic analysis to determine that no memory mappings are placed at an explicit and consistent address. The method of doing so varies per platform. For those platforms requiring the same application running on two different systems, the evaluator may alternatively use the same device. After collecting the first instance of mappings, the evaluator must uninstall the application, reboot the device, and reinstall the application to collect the second instance of mappings.

Platforms: Android....

The evaluator shall run the same application on two different Android systems. Both devices do not need to be evaluated, as the second device is acting only as a tool. Connect via ADB and inspect /proc/PID/maps. Ensure the two different instances share no memory mappings made by the application at the same location.

Platforms: Microsoft Windows....

The evaluator shall run the same application on two different Windows systems and run a tool that will list all memory mapped addresses for the application. The evaluator shall then verify the two different instances share no mapping locations. The Microsoft SysInternals tool, VMMap, could be used to view memory addresses of a running application. The evaluator shall use a tool such as Microsoft's BinScope Binary Analyzer to confirm that the application has ASLR enabled.

Platforms: Apple iOS....

The evaluator shall perform a static analysis to search for any mmap calls (or API calls that call mmap), and ensure that no arguments are provided that request a mapping at a fixed address.

Platforms: Linux....

The evaluator shall run the same application on two different Linux systems. The evaluator shall then compare their memory maps using pmap -x PID to ensure the two different instances share no mapping locations.

Platforms: Oracle Solaris....

The evaluator shall run the same application on two different Solaris systems. The evaluator shall then compare their memory maps using pmap -x PID to ensure the two different instances share no mapping locations.

Platforms: Apple macOS....

The evaluator shall run the same application on two different Mac systems. The evaluator shall then compare their memory maps using vmmap PID to ensure the two different instances share no mapping locations.

The TOE primarily is a Java application that relies on Java-level protection including RuntimeExceptions for Java's memory management. Since Java manages and actively monitors its own memory and does not present the option of opting out of these protections, the Java platform does not have the ability to allocate memory at a fixed address.

To account for native components of the TOE, the evaluator subjected the TOE against the Linux platform requirements above. The evaluator obtained copies of the memory maps form two separate Linux systems running the TOE. Comparing the memory maps, the evaluator found that the memory was sufficiently randomized.

2.5.1.2 ASPP14:FPT_AEX_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator shall verify that no memory mapping requests are made with write and execute permissions. The method of doing so varies per platform.

Platforms: Android....

The evaluator shall perform static analysis on the application to verify that

o mmap is never invoked with both the PROT_WRITE and PROT_EXEC permissions, and

o mprotect is never invoked.

Platforms: Microsoft Windows....

The evaluator shall use a tool such as Microsoft's BinScope Binary Analyzer to confirm that the application passes the NXCheck. The evaluator may also ensure that the /NXCOMPAT flag was used during compilation to verify that DEP protections are enabled for the application.

Platforms: Apple iOS....

The evaluator shall perform static analysis on the application to verify that mprotect is never invoked with the PROT_EXEC permission.

Platforms: Linux....

The evaluator shall perform static analysis on the application to verify that both

o mmap is never be invoked with both the PROT_WRITE and PROT_EXEC permissions, and

o mprotect is never invoked with the PROT_EXEC permission.

Platforms: Oracle Solaris....

The evaluator shall perform static analysis on the application to verify that both

o mmap is never be invoked with both the PROT_WRITE and PROT_EXEC permissions, and

o mprotect is never invoked with the PROT_EXEC permission.

Platforms: Apple macOS....

The evaluator shall perform static analysis on the application to verify that mprotect is never invoked with the PROT_EXEC permission.

The evaluator performed a static analysis and found no evidence that mmap was ever invoked with both the PROT_WRITE and PROT_EXEV permissions or that mprotect was ever invoked with the PROT_EXEC.

2.5.1.3 ASPP14:FPT_AEX_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator shall configure the platform in the ascribed manner and carry out one of the prescribed tests:

Platforms: Android....

Applications running on Android cannot disable Android security features, therefore this requirement is met and no evaluation activity is required.

Platforms: Microsoft Windows....

If the OS platform supports Windows Defender Exploit Guard (Windows 10 version 1709 or later), then the evaluator shall ensure that the application can run successfully with Windows Defender Exploit Guard Exploit Protection configured with the following minimum mitigations enabled; Control Flow Guard (CFG), Randomize memory allocations (Bottom-Up ASLR), Export address filtering (EAF), Import address filtering (IAF), and Data Execution Prevention (DEP). The following link describes how to enable Exploit Protection, https://docs.microsoft.com/en-us/windows/security/threatprotection/windows-defender-exploit-guard/customize-exploit-protection.

If the OS platform supports the Enhanced Mitigation Experience Toolkit (EMET) which can be installed on Windows 10 version 1703 and earlier, then the evaluator shall ensure that the application can run successfully with EMET configured with the following minimum mitigations enabled; Memory Protection Check, Randomize memory allocations (Bottom-Up ASLR), Export address filtering (EAF), and Data Execution Prevention (DEP).

Platforms: Apple iOS....

Applications running on iOS cannot disable security features, therefore this requirement is met and no evaluation activity is required.

Platforms: Linux....

The evaluator shall ensure that the application can successfully run on a system with either SELinux or AppArmor enabled and in enforce mode.

Platforms: Oracle Solaris....

The evaluator shall ensure that the application can run with Solaris Trusted Extensions enabled and enforcing.

Platforms: Apple macOS....

The evaluator shall ensure that the application can successfully run on macOS without disabling any security features.

The evaluator ran the TOE on a platform that had SELinux enabled and enforcing and found no issue with TOE functionality.

2.5.1.4 ASPP14:FPT_AEX_EXT.1.4

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator shall run the application and determine where it writes its files. For files where the user does not choose the destination, the evaluator shall check whether the destination directory contains executable files. This varies per platform:

Platforms: Android....

The evaluator shall run the program, mimicking normal usage, and note where all user-modifiable files are written. The evaluator shall ensure that there are no executable files stored under /data/data/package/ where package is the Java package of the application.

Platforms: Microsoft Windows....

For Windows Universal Applications the evaluator shall consider the requirement met because the platform forces applications to write all data within the application working directory (sandbox). For Windows Desktop Applications the evaluator shall run the program, mimicking normal usage, and note where all user-modifiable files are written. The evaluator shall ensure that there are no executable files stored in the same directories to which the application wrote user-modifiable files.

Platforms: Apple iOS....

The evaluator shall consider the requirement met because the platform forces applications to write all data within the application working directory (sandbox).

Platforms: Linux....

The evaluator shall run the program, mimicking normal usage, and note where all user-modifiable files are written. The evaluator shall ensure that there are no executable files stored in the same directories to which the application wrote user-modifiable files.

Platforms: Oracle Solaris....

The evaluator shall run the program, mimicking normal usage, and note where all user-modifiable files are written. The evaluator shall ensure that there are no executable files stored in the same directories to which the application wrote user-modifiable files.

Platforms: Apple macOS....

The evaluator shall run the program, mimicking normal usage, and note where all user-modifiable files are written. The evaluator shall ensure that there are no executable files stored in the same directories to which the application wrote user-modifiable files.

The evaluator obtained a list of where the TOE wrote files and generated a list of user-writeable files. The evaluator found that the TOE did not store user-writeable files in any of its locations.

2.5.1.5 ASPP14:FPT_AEX_EXT.1.5

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator will inspect every native executable included in the TOE to ensure that stack-based buffer overflow protection is present.

Platforms: Microsoft Windows....

Applications that run as Managed Code in the .NET Framework do not require these stack protections. Applications developed in Object Pascal using the Delphi IDE compiled with RangeChecking enabled comply with this element. For other code, the evaluator shall review the TSS and verify that the /GS flag was used during compilation. The evaluator shall run a tool like, BinScope, that can verify the correct usage of /GS.

For PE , the evaluator will disassemble each and ensure the following sequence appears:

```
mov rcx, QWORD PTR [rsp+(...)]

xor rcx, (...)

call (...)
```

For ELF executables, the evaluator will ensure that each contains references to the symbol _stack_chk_fail.

Tools such as Canary Detector may help automate these activities.

The TOE is primarily a Java application and this test only applies to the native components included in the TOE. The TOE is bundled with a small number of native executables including PE and ELF executables. The evaluator used the recommended Canary Detector tool to verify the _stack_chk_fail symbol and the PE stack canary disassembled sequence in each of the native executables.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.5.2 Use of Supported Services and APIs (ASPP14:FPT_API_EXT.1)

2.5.2.1 ASPP14:FPT_API_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify that the TSS lists the platform APIs used in the application.

Section 6.5 of the ST contains a list of various APIs used in the TOE application.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The evaluator shall then compare the list with the supported APIs (available through e.g. developer accounts, platform developer groups) and ensure that all APIs listed in the TSS are supported.

The evaluator pulled this of supported APIs used by the TOE from the ST. The evaluator compared the list with online documentation for the Open Java Development Kit (OpenJDK) and found references to all of the included APIs.

2.5.3 Software Identification and Versions (ASPP14:FPT_IDV_EXT.1)

2.5.3.1 ASPP14:FPT_IDV_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: If 'other version information' is selected the evaluator shall verify that the TSS contains an explanation of the versioning methodology.

Section 6.5 of the TSS states that the TOE is versioned with a major and minor version number, as well as a build number that is incremented with every update to the TOE.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The evaluator shall install the application, then check for the / existence of version information. If SWID tags is selected the evaluator shall check for a .swidtag file. The evaluator shall open the file and verify that is contains at least a SoftwareIdentity element and an Entity element.

This is tested in conjunction with FPT_TUD_EXT.1-t1 where the evaluator checked the current version as well as for any available updates. During this test, the evaluator observed the version of the TOE.

2.5.4 Use of Third Party Libraries (ASPP14:FPT_LIB_EXT.1)

2.5.4.1 ASPP14:FPT_LIB_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The evaluator shall install the application and survey its installation directory for dynamic libraries. The evaluator shall verify that libraries found to be packaged with or employed by the application are limited to those in the assignment.

The evaluator inventories the location of where the application was installed for files with extensions that matched dynamic libraries including .so, .a, .dll, .la, .jar, and .java. The resulting files were then individually identified to their corresponding library in the ST and ensured that no additional libraries were included.

2.5.5 Integrity for Installation and Update (ASPP14:FPT_TUD_EXT.1)

2.5.5.1 ASPP14:FPT_TUD_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: The evaluator shall check to ensure the guidance includes a description of how updates are performed.

Section 4.8 Updating TNMS Server of the AGD contains details of how updates should be performed.

Testing Assurance Activities: The evaluator shall check for an update using procedures described in either the application documentation or the platform documentation and verify that the application does not issue an error. If it is updated or if it reports that no update is available this requirement is considered to be met.

The evaluator followed the procedures in the AGD to check for an update. The update script reported back the current version, the latest version, and whether the client was considered up to date. Since the client was up to date and no error was issued, this requirement was considered to be met.

2.5.5.2 ASPP14:FPT_TUD_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: The evaluator shall verify guidance includes a description of how to query the current version of the application.

Section 4.7 Checking the Installed Version contains information about how the end-user can check if there are any updates available. The same procedure for checking for updates also will report back the current version of the product and if it is up to date.

Testing Assurance Activities: The evaluator shall query the application for the current version of the software according to the operational user guidance. The evaluator shall then verify that the current version matches that of the documented and installed version.

The evaluator tested this in conjunction with FPT_TUD_EXT.1.1-t1. The evaluator verified the installed and documented versions are the same.

2.5.5.3 ASPP14:FPT_TUD_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator shall verify that the application's executable files are not changed by the application.

Platforms: Apple iOS: The evaluator shall consider the requirement met because the platform forces applications to write all data within the application working directory (sandbox).

For all other platforms, the evaluator shall perform the following test:

Test 1: The evaluator shall install the application and then locate all of its executable files. The evaluator shall then, for each file, save off either a hash of the file or a copy of the file itself. The evaluator shall then run the application and exercise all features of the application as described in the ST. The evaluator shall then compare each executable file with the either the saved hash or the saved copy of the files. The evaluator shall verify that these are identical.

The evaluator installed the application and recorded the timestamps and MD5 hash of all the executables bundled with the TOE. The evaluator then ran the application and ensured functionality. The evaluator then compared the current executable attributes with the previous record and found that they were the same

2.5.5.4 ASPP14:FPT_TUD_EXT.1.4

TSS Assurance Activities: The evaluator shall verify that the TSS identifies how updates to the application are signed by an authorized source. The definition of an authorized source must be contained in the TSS. The evaluator shall also ensure that the TSS (or the operational guidance) describes how candidate updates are obtained.

Section 6.5 of the ST contains information regarding TOE updates. As stated in this section, the vendor packages updates to the TOE in an RPM format and relies upon the Red Hat Enterprise Linux or CentOS operating system to

verify the installation package's signature before installing. Updates are signed by Infinera's GPG key which is provided by the vendor to be installed on the platform.

The same section also identifies the following for how candidate updates are obtained: Updates are obtained through Infinera's Customer Service Portal (https://support.infinera.com/). Updates are in the same format as initial initializations and are installed using the same process as the initial installations.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.5.5 ASPP14:FPT_TUD_EXT.1.5

TSS Assurance Activities: The evaluator shall verify that the TSS identifies how the application is distributed.

Section 6.5 of the ST contains the following about how the TOE is distributed: The vendor packages updates to the TOE in an RPM format and relies upon the Red Hat Enterprise Linux or CentOS operating system to verify the installation package's signature before installing. Updates are signed by Infinera's GPG key which is provided by the vendor to be installed on the platform. Updates are obtained through Infinera's Customer Service Portal (https://support.infinera.com/). Updates are in the same format as initial initializations and are installed using the same process as the initial installations.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: If 'with the platform' is selected the evaluated shall perform a clean installation or factory reset to confirm that TOE software is included as part of the platform OS. If 'as an additional package' is selected the evaluator shall perform the tests in FPT_TUD_EXT.2.

The TOE claims 'as an additional package' and therefore FPT_TUD_EXT.2 is included and no additional testing activity is performed.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.5.6 Integrity for Installation and Update - per TD0664 (ASPP14:FPT_TUD_EXT.2)

2.5.6.1 ASPP14:FPT_TUD_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: If a container image is claimed the evaluator shall verify that application updates are distributed as container images.

If the format of the platform-supported package manager is claimed, the evaluator shall verify that application updates are distributed in the format supported by the platform. This varies per platform:

Platforms: Android....

The evaluator shall ensure that the application is packaged in the Android application package (APK) format.

Platforms: Microsoft Windows....

The evaluator shall ensure that the application is packaged in the standard Windows Installer (.MSI) format, the Windows Application Software (.EXE) format signed using the Microsoft Authenticode process, or the Windows Universal Application package (.APPX) format. See https://msdn.microsoft.com/enus/library/ms537364(v=vs.85).aspx for details regarding Authenticode signing.

Platforms: Apple iOS....

The evaluator shall ensure that the application is packaged in the IPA format.

Platforms: Linux....

The evaluator shall ensure that the application is packaged in the format of the package management infrastructure of the chosen distribution. For example, applications running on Red Hat and Red Hat derivatives shall be packaged in RPM format. Applications running on Debian and Debian derivatives shall be packaged in DEB format.

Platforms: Oracle Solaris....

The evaluator shall ensure that the application is packaged in the PKG format.

Platforms: Apple macOS....

The evaluator shall ensure that application is packaged in the DMG format, the PKG format, or the MPKG format. (TD0664 applied)

The TOE is provided in the format of the platform-supported package manager. As a result, the TOE was provided to the evaluator in the form of an RPM package for all of testing.

2.5.6.2 ASPP14:FPT_TUD_EXT.2.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: Platforms: Android....

The evaluator shall consider the requirement met because the platform forces applications to write all data within the application working directory (sandbox).

Platforms: Apple iOS....

The evaluator shall consider the requirement met because the platform forces applications to write all data within the application working directory (sandbox).

All Other Platforms...

The evaluator shall record the path of every file on the entire filesystem prior to installation of the application, and then install and run the application. Afterwards, the evaluator shall then uninstall the application, and compare the resulting filesystem to the initial record to verify that no files, other than configuration, output, and audit/log files, have been added to the filesystem.

The evaluator recorded every file on the filesystem prior to the installation of the application, and then installed and ran the application. The evaluator then followed AGD guidance to uninstall the TOE and compared the resulting filesystem to the initial record. The only differences in the file scans were related to things not attributable to the TOE or intentionally left behind as they were considered configuration/log files which are allowed per this testing requirement.

2.5.6.3 ASPP14:FPT_TUD_EXT.2.3

TSS Assurance Activities: The evaluator shall verify that the TSS identifies how the application installation package is signed by an authorized source. The definition of an authorized source must be contained in the TSS.

Section 6.5 of the ST states the vendor packages updates to the TOE in an RPM format and relies upon the Red Hat Enterprise Linux or CentOS operating system to verify the installation package's signature before installing. Updates are signed by Infinera's GPG key which is provided by the vendor to be installed on the platform. Updates are obtained through Infinera's Customer Service Portal (https://support.infinera.com/). Updates are in the same format as initial initializations and are installed using the same process as the initial installations.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.6 TRUSTED PATH/CHANNELS (FTP)

2.6.1 PROTECTION OF DATA IN TRANSIT - PER TD0655 (ASPP14:FTP_DIT_EXT.1)

2.6.1.1 ASPP14:FTP_DIT_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: For platform-provided functionality, the evaluator shall verify the TSS contains the calls to the platform that TOE is leveraging to invoke the functionality.

Not applicable, the TOE does not invoke platform-provided functionality.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The evaluator shall perform the following tests.

Test 1: The evaluator shall exercise the application (attempting to transmit data; for example by connecting to remote systems or websites) while capturing packets from the application. The evaluator shall verify from the packet capture that the traffic is encrypted with HTTPS, TLS, DTLS, SSH, or IPsec in accordance with the selection in the ST.

Test 2: The evaluator shall exercise the application (attempting to transmit data; for example by connecting to remote systems or websites) while capturing packets from the application. The evaluator shall review the packet capture and verify that no sensitive data is transmitted in the clear.

Test 3: The evaluator shall inspect the TSS to determine if user credentials are transmitted. If credentials are transmitted the evaluator shall set the credential to a known value. The evaluator shall capture packets from the application while causing credentials to be transmitted as described in the TSS. The evaluator shall perform a string search of the captured network packets and verify that the plaintext credential previously set by the evaluator is not found.

Platforms: Android....

If 'not transmit any data' is selected, the evaluator shall ensure that the application's AndroidManifest.xml file does not contain a uses-permission or uses-permission-sdk-23 tag containing android:name='android.permission.INTERNET'. In this case, it is not necessary to perform the above Tests 1, 2, or 3, as the platform will not allow the application to perform any network communication.

Platforms: Apple iOS....

If 'encrypt all transmitted data' is selected, the evaluator shall ensure that the application's Info.plist file does not contain the NSAllowsArbitraryLoads or NSExceptionAllowsInsecureHTTPLoads keys, as these keys disable iOS's Application Transport Security feature.

Test 1 – the evaluator reanalyzed the packet capture from FDP_NET_EXT.1-t1. The TOE claims to encrypt all sensitive data. All data transmitted by the TOE is either encrypted with TLS (for communicating with the external TNMS Client) SSH (for communicating with external network elements), or is not considered sensitive data (as is the case with version information to the vendor's public versioning server). The evaluator did not observe any undocumented traffic form the TOE.

Test 2 – The evaluator reanalyzed the packet capture from FDP_NET_EXT.1-t1. The evaluator attempted to search for the username, password, and version information used by the external TNMS Client in plaintext form and could not find them. The evaluator found no other sensitive data in packet capture

Test 3 – The TOE does receive credentials from the external TNMS Client. The evaluator tested this alongside test 2 and found that the credentials were not present in plaintext in the packet capture.

3. Protection Profile SAR Assurance Activities

The following sections address assurance activities specifically defined in the ASPP14/SSH10/PKGTLS11 that correspond with Security Assurance Requirements.

3.1 Development (ADV)

3.1.1 Basic Functional Specification (ADV_FSP.1)

Assurance Activities: There are no specific assurance 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, 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 assurance 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.

The assurance activities from section 5.1 of ASPP14 have been performed and the analysis of the evaluator is documented in the previous sections of this document.

3.2 Guidance documents (AGD)

3.2.1 Operational User Guidance (AGD_OPE.1)

Assurance Activities: Some of the contents of the operational guidance will be verified by the assurance activities in Section 5.1 and evaluation of the TOE according to the [CEM]. The following additional information is also required. If cryptographic functions are provided by the TOE, the operational guidance shall contain instructions for configuring the cryptographic engine associated with the evaluated configuration of the TOE. It shall provide a warning to the administrator that use of other cryptographic engines was not evaluated nor tested during the CC evaluation of the TOE. The documentation must describe the process for verifying updates to the TOE by verifying a digital signature - this may be done by the TOE or the underlying platform. The evaluator shall 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 TOE (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 TOE will likely contain security functionality that does not fall in the scope of evaluation under this PP. The operational guidance shall make it clear to an administrator which security functionality is covered by the evaluation activities.

Section 4.4 Enabling FIPS mode of the AGD indicates that the TOE installation requires setting up the cryptographic library into FIPS mode in order to be in compliance with the Common Criteria evaluated configuration. Additionally, this section also states that “the use of any other cryptographic engines, or configurations other than what is described in this section was not evaluated nor tested during the Common Criteria Evaluation of TNMS Server

Section 4.8 Updating TNMS Server contains all required information including where to obtain new updates, instructions to update the TOE, verifying signatures, and determining if it was successful. Updates are obtained

from Infinera's Customer Service portal. This section states that updates are installed in the same manner as the initial installation which is described in *Section 4.1 TNMS Server Installation*. In that referenced section, any necessary instructions for making the installation accessible to the platform are included. Similarly, the success of the update can be checked in the same manner as the initial update, by "check[ing] the status of TNMS Server services as described in 4.3 - Managing TNMS Server Services. All services should have status RUNNING." The signature is verified by the platform rpm tool using the TNMS GPG public key used during installation.

The TOE does not contain any obvious functionality that is not included under the scope of this evaluation aside from any mention of the external TNMS Client. *Section 1 Introduction* states "The full TNMS system consists of a TNMS server and a TNMS client. This guidance document covers the TNMS Server only. The TNMS Client is not included as it is being evaluated separately. This document is provided as a supplement to the TNMS Customer Documentation provided with every TNMS installation and describes how to install and configure the TNMS Server Component as the evaluated configuration compliant with the Common Criteria for Information Technology Security Evaluation version 3.1."

3.2.2 Preparative Procedures (AGD_PRE.1)

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

The ST claims that TOE is a Java application capable of running on a Linux environment with either Red Hat Enterprise Linux (RHEL) or CentOS 7.9 with Amazon Corretto (OpenJDK) JDK/JRE 11.0.6 pre-installed. The AGD is in agreement with this claim as the list of supported OS's listed in *Section 2.1 Supported Operating Systems* and the list of software prerequisites in *Section 3.3 Software Pre-Requisites* identify that the TOE runs on either Linux platform and iterates through the software pre-requisites including Amazon Corretto JDK 11.0.6.10.1 for Linux

3.3 Life-cycle support (ALC)

3.3.1 Labelling of the TOE (ALC_CMC.1)

Assurance Activities: The evaluator shall 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 shall check the AGD guidance and TOE 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 TOE, the evaluator shall examine the information on the web site to ensure that the information in the ST is sufficient to distinguish the product.

The title of the AGD document specifies a specific version of the TNMS Server and the evaluator ensure that was consistent with the samples received during testing and the version contained in the ST.

3.3.2 TOE CM Coverage (ALC_CMS.1)

Assurance Activities: 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 TOE is specifically identified and that this identification is consistent in the ST and in the AGD guidance (as done in the assurance 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 shall 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 shall provide information on how to configure the environment to ensure that buffer overflow protection mechanisms in the environment(s) are invoked (e.g., compiler flags). The evaluator shall ensure that this documentation also includes an indication of whether such protections are on by default, or have to be specifically enabled. The evaluator shall 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.

The evaluator noted that the AGD identifies that the subject of the document is for the matching version in the ST. Included in the subsection *Section 2.1 Supported Operating Systems,* the AGD states the TNMS Server is primarily a Java application and features some native level libraries. For all components, the TNMS server is compiled with all necessary compilation flags to ensure that all required environmental protections are enabled by default and require no further configuration under Linux.

3.3.3 Timely Security Updates (ALC_TSU_EXT.1)

Assurance Activities: The evaluator shall 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 shall verify that this description addresses the entire application.

The evaluator shall also verify that, in addition to the TOE developer's process, any third-party processes are also addressed in the description. The evaluator shall also verify that each mechanism for deployment of security updates is described. The evaluator shall 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 TOE patching this vulnerability, to include any third-party or carrier delays in deployment. The evaluator shall verify that this time is expressed in a number or range of days. The evaluator shall verify that this description includes the publicly available mechanisms (including either an email address or website) for reporting security issues related to the TOE.

The evaluator shall 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.

Section 6.6 of the ST claims the vendor provides timely security updates for the TOE and third party libraries included in the TOE. The vendor aims for updates as soon as possible with a maximum of 30 days. Updates are packaged in the same RPM format as the original installation of the TOE. The vendor maintains a customer portal where updates can be found and new issues can be reported.

3.4 Tests (ATE)

3.4.1 Independent Testing - Conformance (ATE_IND.1)

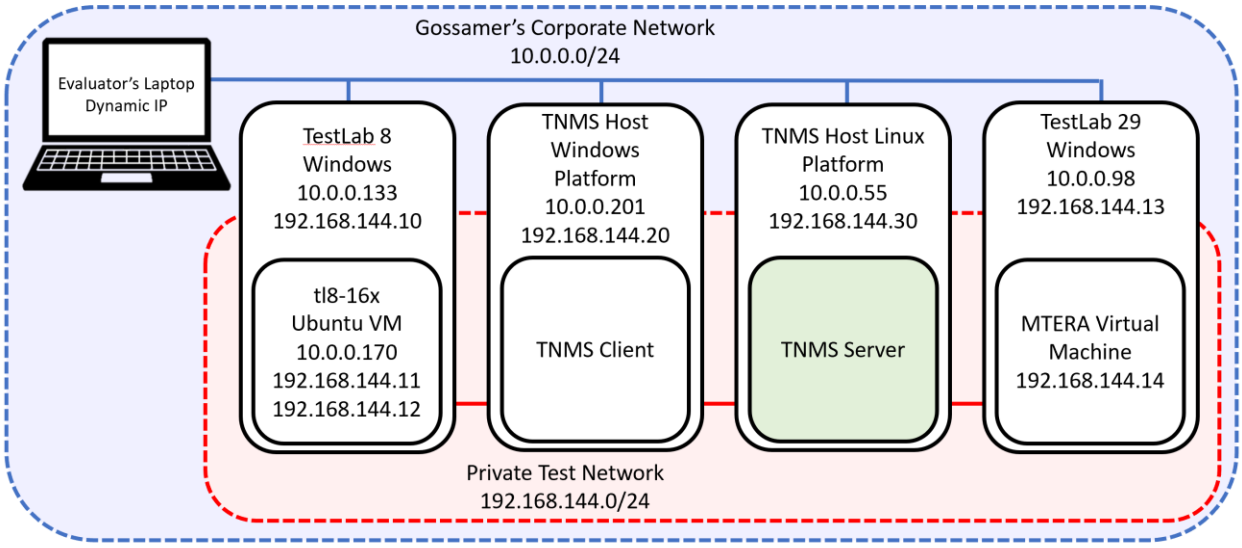
Assurance Activities: The evaluator shall prepare a test plan and report documenting the testing aspects of the system, including any application crashes during testing. The evaluator shall determine the root cause of any application crashes and include that information in the report. The test plan covers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

While it is not necessary to have one test case per test listed in an Assurance 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 TOE 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, SSH). 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 shall 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.

The evaluator created a proprietary Detailed Test Report (DTR) to address all aspects of this requirement. The DTR discusses the test configuration, test cases, expected results, and test results. The following diagram depicts the test environment



3.5 Vulnerability assessment (AVA)

3.5.1 Vulnerability Survey (AVA_VAN.1)

Assurance Activities: The evaluator shall 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 shall also run a virus scanner with the most current virus definitions against the application files and verify that no files are flagged as malicious. 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.

The virus definition search and vulnerability analysis are in the proprietary Detailed Test Report (DTR) prepared by the evaluator. The evaluator ran a Windows Defender virus scan with current virus definitions against all of the TOE components and verified that no files are flagged as malicious. The vulnerability analysis includes a public search for vulnerabilities. The evaluator searched on 12/01/2022 the National Vulnerability Database (NVD) from the NIST website and the Vulnerability Notes Database (VND) from the CERT Knowledgebase using the following terms:

- Infinera
- Infinera Corporation
- Transcend Network Management System
- TNMS
- ASM
- AXL Software Radius
- Angular
- Animal Sniffer Annotations
- Apache ActiveMQ
- Apache Commons
- Apache Groovy
- Apache HttpClient

- Apache HttpComponents Core
- Apache Kafka
- Apache Log4j
- Apache Tomcat
- Apache XML
- Apache Xalan
- Apache Xerces2
- log4j
- AspectJ
- AsyncHttpClient
- AutoValue Annotations
- Avalon Framework
- Bouncy Castle TLS
- Batik
- Bootstrap (Twitter)
- Bouncy Castle
- CDIs
- Castor
- Checker Qual
- Common Annotations
- Dagger
- Db4o
- Disruptor
- Elsa Serialization
- Expression Language
- Extended StAX
- FindBugs jsr305
- GeoServer
- Goldman Sachs
- Gson
- Guice
- Guava
- H2 Database Engine
- Hamcrest
- HawtJNI Runtime
- Hibernate
- HyperSQL Database Engine
- Install Anywhere
- J2EE Connector Architecture
- J2EE Management
- J2ObjC Annotations
- JAVAX RMI
- JAXB Implementation
- JAXB Runtime
- JBoss EJB client
- JBoss Logging
- JBoss Marshalling
- JBoss Remoting
- JBoss classfilewriter
- JCL 1.2 Implemented Over SLF4J
- JDOM
- JFreeChart
- JFreeReport
- JFtp
- JGraphT
- JSON Web Token
- JSch
- JUL to SLF4J bridge
- JUnit
- JZlib
- JacORB
- Jackson
- Jakarta Annotations
- Jakarta JCS
- Jakarta XML
- Jansi
- Java Architecture for XML Binding
- Java Communications
- Java DMK
- Java Mail
- Java Servlet
- Java gRPC
- Java inject from the JSR-330 Expert Group
- Java Interceptors
- JavaBeans Activation
- JavaHelp
- JavaServer Pages TagLib Implementation
- Java Authentication and Authorization Service
- Javassist
- Jcommander
- Jetbrains
- Jline
- Jscape
- Koloboke Collections
- Kotlin Stdlib
- MapStruct
- Metrics Core
- Metrics Integration
- ModeShape
- Monfox TL1
- Netty Project
- OW2 Utilities
- OpenCensus
- OpenFusion CORBA Services
- Openmap
- Oracle Database JDBC Drivers
- Oracle Deinstallation
- Oracle Fast Infoset
- Oracle iStack
- Picocli
- Protocol Buffer Java
- Protocol Buffers
- Quartz
- Querydsl
- Reactor Components
- Remoting-jmx
- SLF4J
- SNMP4J
- SmoothieMap
- SnakeYAML
- Spring
- Spring Boot
- Spring Framework
- Spring Security
- Stax2
- Stormpot
- Super CSV
- TXW Runtime
- Thymeleaf
- Trove for Java
- Tyrex
- Undertow Core
- VT Crypt
- VT Dictionary
- VT Password
- Weld Core
- Wildfly
- Woodstox
- XML Commons External Components XML Extensions
- Xerces
- XmlSchema Core
- aosphere-cdi
- aosphere-runtime

- aosphere-socketio
- cglib
- codegen
- concurrent
- com.typesafe:config
- dom4j
- edtFTPj
- error-prone annotations
- exp4j
- fast-serialization
- google-gson
- image4j
- grpc-context
- istack commons
- jQuery
- jacorb
- javax.ejb
- javax.transaction
- jgraph
- jgraphx
- jmxremote_optional repackaged as module
- jmxtrans
- jt-classbreaks
- kotlin-argparser
- mysema-commons-lang
- level db
- logkit
- net.sourceforge.stream support:java9-concurrent-backport
- objenesis
- ognl
- ojdbc
- ojdbc8
- perfmark
- reflections
- JBoss Marshalling river
- spring-boot-actuator
- swingx
- syslog4j
- thymeleaf-extras-springsecurity4
- thymeleaf-layout-dialect
- thymeleaf-spring4
- ucp
- unbescape
- wasync
- webdavlib
- wildfly-common
- xenocom
- xnio-api.

The public vulnerability search did not return back any unresolved vulnerabilities.