Assurance Activities Report
for
Nessus Network Monitor v6.2.2

Version 1.1
6/28/2023

Evaluated By:

Leidos Inc.
https://www.leidos.com/civil/commercial-cyber/product-compliance
Common Criteria Testing Laboratory
6841 Benjamin Franklin Drive
Columbia, MD 21046

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

The Developer of the TOE:

Tenable, Inc.

6100 Merriweather Drive
12th Floor
Columbia, MD 21044

The TOE Evaluation was Sponsored by:

Tenable, Inc.

6100 Merriweather Drive

12th Floor

Columbia, MD 21044

Evaluation Personnel:
Greg Beaver
Pascal Patin
Srilekha Vangal
Armin Najafabadi

Common Criteria Version:

- Common Criteria for Information Technology Security Evaluation Part 1: Introduction and general model, 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 Version:

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

Protection Profiles:

- Protection Profile for Application Software, Version 1.4, 7 October 2021 [PP_APP_v1.4]
- Functional Package for Transport Layer Security (TLS), Version 1.1, 1 March 2019 [PKG_TLS_V1.1]

Revision History

Version	Date	Description
0.1	2/8/2023	Initial draft
1.0	6/1/2023	Released version
1.1	6/28/2023	Updated for validator ECR comments

Contents

1. INTRODUCTION 1

1.1 TECHNICAL DECISIONS 1
1.2 SAR EVALUATION 2
1.3 REFERENCES 3

2. SECURITY FUNCTIONAL REQUIREMENT EVALUATION ACTIVITIES 4

2.1 CRYPTOGRAPHIC SUPPORT (FCS) 4
2.1.1 FCS_CKM.1 Cryptographic Asymmetric Key Generation 4
2.1.2 FCS_CKM.1/AK Cryptographic Asymmetric Key Generation 5
2.1.3 FCS_CKM.1/PBKDF Password Conditioning 8
2.1.4 FCS_CKM.2 Cryptographic Key Establishment 8
2.1.5 FCS_COP.1/SKC Cryptographic Operation - Encryption/Decryption 11
2.1.6 FCS_COP.1/Hash Cryptographic Operation - Hashing 17
2.1.7 FCS_COP.1/Sig Cryptographic Operation - Signing 18
2.1.8 FCS_COP.1/KeyedHash Cryptographic Operation - Keyed-Hash Message Authentication 19
2.1.9 FCS_HTTPS_EXT.1/Server HTTPS Protocol 19
2.1.10 FCS_HTTPS_EXT.2 HTTPS Protocol with Mutual Authentication 20
2.1.11 FCS_RBG_EXT.1 Random Bit Generation Services 20
2.1.12 FCS_RBG_EXT.2 Random Bit Generation from Application 22
2.1.13 FCS_STO_EXT.1 Storage of Credentials 24
2.1.14 FCS_TLS_EXT.1 TLS Protocol (TLS Package) 24
2.1.15 FCS_TLSS_EXT.1 TLS Server Protocol (TLS Package) 25
2.1.16 FCS_TLSS_EXT.2 TLS Server Support for Mutual Authentication (TLS Package) 29
2.2 USER DATA PROTECTION (FDP) 32
2.2.1 FDP_DAR_EXT.1(1) Encryption Of Sensitive Application Data 32
2.2.2 FDP_DAR_EXT.1(2) Encryption Of Sensitive Application Data 33
2.2.3 FDP_DEC_EXT.1 Access to Platform Resources 34
2.2.4 FDP_NET_EXT.1 Network Communications 36
2.3 IDENTIFICATION AND AUTHENTICATION (FIA) 37
2.3.1 FIA_X509_EXT.1 X.509 Certificate Validation 37
2.3.2 FIA_X509_EXT.2 X.509 Certificate Authentication 40
2.4 SECURITY MANAGEMENT (FMT) 41
2.4.1 FMT_CFG_EXT.1 Secure by Default Configuration 41
2.4.2 FMT_MEC_EXT.1 Supported Configuration Mechanism 43
2.4.3 FMT_SMF.1 Specification of Management Functions 44
2.5 PRIVACY (FPR) 45
2.5.1 FPR_ANO_EXT.1 User Consent for Transmission of Personally Identifiable Information 45
2.6 PROTECTION OF THE TSF (FPT) 45
2.6.1 FPT_AEX_EXT.1 Anti-Exploitation Capabilities 45
2.6.2 FPT_API_EXT.1 Use of Supported Services and APIs 49
2.6.3 FPT_LIB_EXT.1 Use of Third Party Libraries 49
2.6.4 FPT_IDV_EXT.1 Software Identification and Versions 49
2.6.5 FPT_TUD_EXT.1 Integrity for Installation and UpdateFPT_TUD_EXT.1.1 50
2.6.6 FPT_TUD_EXT.2 Integrity for Installation and Update 52
2.7 TRUSTED PATH/CHANNELS (FTP) 54
2.7.1 FTP_DIT_EXT.1 Protection of Data in Transit 54

3. SECURITY ASSURANCE REQUIREMENT ASSURANCE ACTIVITIES 56

3.1 DEVELOPMENT (ADV) 56
3.1.1 Basic Functional Specification (ADV_FSP.1) 56
3.2 GUIDANCE DOCUMENTS (AGD) 56
3.2.1 Operational User Guidance (AGD_OPE.1) 56
3.2.2 Preparative Procedures (AGD_PRE.1) 57
3.3 TESTS (ATE) 57
3.3.1 Independent Testing – Conformance (ATE_IND.1) 57
3.4 VULNERABILITY ASSESSMENT (AVA) 58
3.4.1 Vulnerability Survey (AVA_VAN.1) 58
3.5 LIFE-CYCLE SUPPORT (ALC) 61
3.5.1 Labeling of the TOE (ALC_CMC.1) 61
3.5.2 TOE Coverage (ALC_CMS.1) 62
3.5.3 Timely Security Update (ALC_TSU_EXT.1) 62

LIST OF TABLES

NO TABLE OF FIGURES ENTRIES FOUND.

1. Introduction

This document presents the results of performing assurance activities associated with the Nessus Network Monitor v6.2.2 evaluation. This report contains sections documenting the performance of evaluation activities associated with each of the Security Functional Requirements (SFRs) and Security Assurance Requirements (SARs) as specified in the following document:

- *Protection Profile for Application Software*, Version 1.4, 7 October 2021 [PP_APP_v1.4]
- *Functional Package for Transport Layer Security (TLS)*, Version 1.1, 1 March 2019 [PKG_TLS_V1.1]

Note that, in accordance with NIAP Policy Letter #5, all cryptography in the TOE for which NIST provides validation testing of FIPS-approved and NIST-recommended cryptographic algorithms and their individual components must be NIST validated. The CCTL will verify that the claimed NIST validation complies with the NIAP-approved PP requirements the TOE claims to satisfy. The CCTL verification of the NIST validation will constitute performance of the associated assurance activity. As such, Test assurance activities associated with functional requirements within the scope of Policy Letter #5 are performed by verification of the relevant CAVP certification and not through performance of any testing as specified in the claimed PP documents.

Testing of the TOE was performed from October 2022 to June 2023.

1.1 Technical Decisions

This subsection lists the Technical Decisions that have been issued by NIAP against [PP_APP_v1.4], along with rationale as to their applicability or otherwise to this evaluation.

TD0743: FTP_DIT_EXT.1.1 Selection exclusivity

- The TD is applicable to the evaluation. The ST accounts for this TD.

TD0736: Number of elements for iterations of FCS_HTTPS_EXT.1

- The TD is applicable to the evaluation. The ST and the Test activities have been updated for FCS_HTTPS_EXT.1.3/Server.

TD0719: ECD for PP APP V1.3 and 1.4

- The TD is applicable to the evaluation. The TD does not affect the ST nor any of the evaluation activities. The TD provides formal definitions for its extended components.

TD0717: Format changes for PP_APP_V1.4

- The TD is applicable to the evaluation. The TD affects FCS_CKM.1, FCS_CKM.2, FCS_CKM.1/AK, FCS_CKM.1/PBKDF, FCS_COP.1/Hash, FCS_COP.1/KeyedHash, FCS_COP.1/Sig, and FCS_COP.1/SKC.

TD0669: FIA_X509_EXT.1 Test 4 Interpretation

- The TD is applicable to the evaluation. The TD affects FIA_X509_EXT.1 Test 4.

TD0664: Testing activity for FPT_TUD_EXT.2.2

- The TD is applicable to the evaluation. Tests for FPT_TUD_EXT.2.2 have been modified.

TD0650: Conformance claim sections updated to allow for MOD_VPNC_V2.3 and 2.4

- The TD is not applicable to the evaluation. No change to ST; affects only PP conformance claims and the ST does not claim conformance to the relevant PP-Module.

TD0628: Addition of Container Image to Package Format

- The TD is applicable to the evaluation. The ST has been updated for FPT_TUD_EXT.2.1. The evaluation activities have been updated for FPT_TUD_EXT.2.1.

TD0624: Addition of DataStore for Storing and Setting Configuration Options

- The TD is not applicable. The TOE is not evaluated on an Android platform.

This subsection lists the Technical Decisions that have been issued by NIAP against [PKG_TLS_V1.1], along with rationale as to their applicability or otherwise to this evaluation.

TD0739: PKG_TLS_V1.1 has 2 different publication dates

- The TD is not applicable to the TOE. The TOE does not implement RSA-based key establishment methods.

TD0726: Corrections to (D)TLSS SFRs in TLS 1.1 FP

- The TD is applicable to the TOE. The ST accounts for this TD in FCS_TLSS_EXT.1.3.

TD0588: Session Resumption Support in TLS package

- The TD is applicable to the TOE. The ST contains FCS_TLSS_EXT.1. The FCS_TLSS_EXT.1.1 Test 4.3 evaluation activity has been modified accordingly.

TD0513: CA Certificate loading

- The TD is not applicable to the evaluation. The TOE does not claim FCS_TLSC_EXT.1.

TD0499: Testing with pinned certificates

- The TD is not applicable to the evaluation. The TOE does not claim FCS_TLSC_EXT.1.

TD0469: Modification of test activity for FCS_TLSS_EXT.1.1 test 4.1

- The TD is applicable to the evaluation. No change to the ST, however the test activity for FCS_TLSS_EXT.1.1 test 4.1 has been modified.

TD0442: Updated TLS Ciphersuites for TLS Package

- The TD is applicable to the evaluation. However, there is no change to ST; affects selections in FCS_TLSS_EXT.1 that are not applicable to the TOE.

1.2 SAR Evaluation

The following Security Assurance Requirements (SARs) were evaluated during the evaluation of the TOE:

SAR	Verdict
ASE_CCL.1	Pass
ASE_ECD.1	Pass
ASE_INT.1	Pass
ASE_OBJ.2	Pass

SAR	Verdict
ASE_REQ.2	Pass
ASE_TSS.1	Pass
ADV_FSP.1	Pass
AGD_OPE.1	Pass
AGD_PRE.1	Pass
ALC_CMC.1	Pass
ALC_CMS.1	Pass
ALC_TST_EXT.1	Pass
ATE_IND.1	Pass
AVA_VAN.1	Pass

The evaluation work units are listed in the proprietary ETR. The evaluators note per the PP evaluation activities that many of the SARs were successfully evaluated through completion of the associated evaluation activities present in the claimed PP and PP-Modules.

1.3 References

[ST] *Nessus Network Monitor 6.2.2 Security Target,* Version 1.1, June 28, 2023

[User] *Tenable Nessus Network Monitor 6.2.x User Guide*, Last Updated: May 18, 2023

[Test] *Tenable Nessus Network Monitor 6.2.2 Common Criteria Test Report and Procedures*, Version 1.1, June 28, 2023

[TUG] Tenable.sc 6.1.x User Guide, Last Revised: March 22, 2023

(Not included in the evaluation. Used for reference only.)

2. Security Functional Requirement Evaluation Activities

This section describes the evaluation activities associated with the SFRs defined in the ST and the results of those activities as performed by the evaluation team. The evaluation activities are derived from [PP_APP_v1.4] and the [PKG_TLS_V1.1]. NIAP Technical Decisions have been applied and are identified as appropriate.

2.1 Cryptographic Support (FCS)

2.1.1 FCS_CKM_EXT.1 Cryptographic Key Generation Services

The following table lists the cryptographic functions supported by the TOE and associated SFRs, the specific algorithms that are claimed for these functions, and the relevant CAVP certificate validation lists and certificate numbers for each.

Functions	Standards	Certificates
FCS_CKM.1/AK Cryptographic Asymmetric Key Generation		
ECC key pair generation (NIST curves P-256, P-384, P-521)	FIPS PUB 186-4	A3617: ECDSA KeyGen (FIPS186-4)
FCS_CKM.2 Cryptographic Key Establishment		
ECDSA based key establishment	NIST SP 800-56A	A3617: KAS-ECC-SSC Sp800-56Ar3
FCS_COP.1/SKC Cryptographic Operation – Encryption/Decryption		
AES-CBC (128, 256 bits)	CBC as defined in NIST SP 800-38A	A3617: AES-CBC
AES-GCM (128, 256 bits)	GCM as defined in NIST SP 800-38D	A3617: AES-GCM
AES-XTS (256 bits)	XTS as defined in NIST SP 800-38E	A3617: AES-XTS
FCS_COP.1/Hash Cryptographic Operation – Hashing		
SHA-256, SHA-384, SHA-512 (digest sizes 256, 384, and 512 bits)	FIPS PUB 180-4	A3617: SHA2-256 A3617: SHA2-384 A3617: SHA2-512
FCS_COP.1/Sig Cryptographic Operation – Signing		
RSA (2048-bit or greater)	FIPS PUB 186-4, Section 4	A3617: RSA SigGen (FIPS 186-4) A3617: RSA SigVer (FIPS 186-4)
FCS_COP.1/KeyedHash Cryptographic Operation – Keyed Hash Message Authentication		
HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512	FIPS PUB 198-1 FIPS PUB 180-4	A3617: HMAC-SHA2-256 A3617: HMAC-SHA2-384 A3617: HMAC-SHA2-512
FCS_RBG_EXT.2 Random Bit Generation from Application		
CTR_DRBG DRBG (256 bits)	NIST SP 800-90A NIST SP 800-57	A3617: Counter DRBG

2.1.1.1 TSS Evaluation Activity

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.

The SFR states that the TOE "implements asymmetric key generation". [ST] Section 6.2 states that the TOE uses a NIST-validated implementation to generate asymmetric keys in support of TLS communications.

2.1.1.2 Guidance Evaluation Activity

None.

2.1.1.3 Test Evaluation Activity

None.

2.1.2 FCS_CKM.1/AK Cryptographic Asymmetric Key Generation

2.1.2.1 TSS Evaluation Activity

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.

The SFR states that the TOE implements cryptographic asymmetric key generation. [ST] Section 6.2 states that the TOE uses a NIST-validated implementation to generate asymmetric keys in support of TLS communications. The TOE implements ECC key pair generation using NIST curves P-256, P-384, and P-521.

2.1.2.2 Guidance Evaluation Activity

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 guidance provided by [User] includes instructions to the administrator on how to configure the TOE so that it uses the ECC key generation schemes in support of the specified TLS ciphersuites. Refer to "Additional Resources > Configure NNM for NIAP Compliance".

2.1.2.3 Test Evaluation Activity

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:

$n = p \cdot q$,

p and q are probably prime according to Miller-Rabin tests,

GCD(p-1,e) = 1,

GCD(q-1,e) = 1,

$2^{16} \leq e \leq 2^{256}$ and e is an odd integer,

$|p-q| > 2^{nlen/2 - 100}$,

$p \geq 2^{nlen/2 - 1/2}$,

$q \geq 2^{nlen/2 - 1/2}$,

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

$e \cdot d = 1 \bmod LCM(p-1,q-1)$.

Key Generation for Elliptic Curve Cryptography (ECC)

FIPS 186-4 ECC Key Generation Test For each supported NIST curve, i.e., P-256, P-384 and P521, 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., P-256, P-384 and P-521, the evaluator shall generate 10 private/public key pairs using the key generation function of a known good implementation and modify five of the public key values so that they are incorrect, leaving five values unchanged (i.e., correct). The evaluator shall obtain in response a set of 10 PASS/FAIL values.

Key Generation for Finite-Field Cryptography (FFC)

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 $1 \leq x \leq q-1$

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

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

$g \neq 0,1$

q divides p-1

$g^q \bmod p = 1$

$g^x \bmod p = y$

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 implements ECC key pair generation (NIST curves P-256, P-384, P-521) according to the FIPS PUB 186-4 Standard. Correct operation was verified by CAVP Certificate A3617.

2.1.3 FCS_CKM.1/PBKDF Password Conditioning

2.1.3.1 TSS Evaluation Activity

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.

[ST] Section 6.2 states that passphrases for certificate encryption are encrypted by the TOE using AES-XTS and administrative credentials to the TOE are encrypted using PBKDF2. The TOE uses the DRBG specified in FCS_RBG_EXT.2 to generate salts that contain at least as many entropy bits as the output key length. The TOE's PBKDF2 implementation performs 10,000 iterations and outputs a 128-bit strength key. Password-based derived keys are formed using a 128-bit salt that is randomly generated by the TOE's DRBG. This is input to the PBKDF function along with the password and specified hashing algorithm, which is SHA-512.

2.1.3.2 Guidance Evaluation Activity

None.

2.1.3.3 Test Evaluation Activity

None.

2.1.4 FCS_CKM.2 Cryptographic Key Establishment

2.1.4.1 TSS Evaluation Activity

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.

The security requirement states that the TOE uses elliptic curve-based key establishment schemes. [ST] Section 6.2 states that the TOE implements the use of elliptic curves as the method of key establishment.

The TSF presents secp256r1, secp384r1 and secp521r1 as the supported values in the Supported Groups extension and uses the same NIST curves for key establishment.

2.1.4.2 Guidance Evaluation Activity

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 guidance provided by [User] includes instructions to the administrator on how to configure the TOE so that it uses the ECC key establishment schemes in support of the specified TLS ciphersuites. Refer to "Additional Resources > Configure NNM for NIAP Compliance".

2.1.4.3 Test Evaluation Activity

Tests

***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-KEMKWS 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 RSAES-PKCS1-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 safeprime groups. This test must be performed for each safe-prime group that each protocol uses.

The TOE implements ECDSA based key establishment according to the NIST SP 800-56A Standard. Correct operation was verified by CAVP Certificate A3617.

2.1.5 FCS_COP.1/SKC Cryptographic Operation - Encryption/Decryption

2.1.5.1 TSS Evaluation Activity

None.

2.1.5.2 Guidance Evaluation Activity

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.

As described in Section 6.2 of [ST] ("Cryptographic Support"), the TOE uses a NIST-validated implementation to perform AES encryption and decryption in support of both TLS communications and secure storage of credentials.

The guidance provided by [User] includes instructions to the administrator on how to configure the TOE to:

- Use the specified TLS cipher suites, which also determines the key sizes and modes for AES the TOE will use
- Convert the TOE database to use the specified AES encryption mode and key size (AES-XTS-256).

Refer to “Additional Resources > Configure NNM for NIAP Compliance”.

2.1.5.3 Test Evaluation Activity

Tests

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 nonzero 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 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 implements AES-CBC (128, 256 bits) as defined in NIST SP 800-38A. The TOE implements AES-GCM (128, 256 bits) as defined in NIST SP 800-38D. The TOE implements AES-XTS (128, 256 bits) as defined in NIST SP 800-38E. The correct operation of the algorithms was verified by CAVP Certificate A3617.

2.1.6 FCS_COP.1/Hash Cryptographic Operation - Hashing

2.1.6.1 TSS Evaluation Activity

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.

Application cryptographic functions that implement the hash function are identified below.

[ST] Section 6.2 states that all digital signatures used for the establishment of TLS communications use 2048-bit RSA.

The TLS client implementations support the following TLS cipher suites in the TOE's evaluated configuration:

- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384

The TOE is packaged as an .rpm file for Linux and an .exe file for Windows. Each are digitally signed by Tenable using 2048-bit RSA.

2.1.6.2 Guidance Evaluation Activity

None.

2.1.6.3 Test Evaluation Activity

Tests

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 implements SHA-256, SHA-384, SHA-512 (digest sizes 256, 384, and 512 bits) according to the FIPS PUB 180-4 Standard. The correct operation of both algorithms was verified by CAVP Certificate A3617.

2.1.7 FCS_COP.1/Sig Cryptographic Operation - Signing

2.1.7.1 TSS Evaluation Activity

None.

2.1.7.2 Guidance Evaluation Activity

None.

2.1.7.3 Test Evaluation Activity

Tests

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 implements RSA Signature Generation and Verification (2048-bit or greater) according to FIPS PUB 186-4, Section 4. Correct operation was verified by CAVP Certificate A3617.

2.1.8 FCS_COP.1/KeyedHash Cryptographic Operation - Keyed-Hash Message Authentication

2.1.8.1 TSS Evaluation Activity

None.

2.1.8.2 Guidance Evaluation Activity

None.

2.1.8.3 Test Evaluation Activity

Tests

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 implements Keyed Hash Message Authentication using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 according to FIPS PUB 198-1 and FIPS PUB 180-4 standards. Correct operation was verified by CAVP Certificate A3617.

2.1.9 FCS_HTTPS_EXT.1/Server HTTPS Protocol

2.1.9.1 TSS Evaluation Activity

FCS_HTTPS_EXT.1.1/Server

The evaluator shall examine the TSS and determine that enough detail is provided to explain how the implementation complies with RFC 2818.

FCS_HTTPS_EXT.1.2/Server

None

FCS_HTTPS_EXT.1.3/Server

None

The TOE uses TLS server functionality for communications between the TOE and the environmental Tenable.sc application and from remote administrators to the Web GUI interface. All uses of the TOE's

TLS server support mutual authentication. The TOE's implementation of HTTPS conforms to RFC 2818. In all cases, the connection will be rejected if certificate validation fails.

2.1.9.2 Guidance Evaluation Activity

None.

2.1.9.3 Test Evaluation Activity

Tests

Modified per TD0736

Other tests are performed in conjunction with the TLS Functional Package, FCS_HTTPS_EXT.2 (dependent on selections in FTP_DIT_EXT.1), and FIA_X509_EXT.1.

N/A. As noted in the Protection Profile there are no tests specifically for this SFR. It is covered by the TLS package.

2.1.10 FCS_HTTPS_EXT.2 HTTPS Protocol with Mutual Authentication

2.1.10.1 TSS Evaluation Activity

None.

2.1.10.2 Guidance Evaluation Activity

None.

2.1.10.3 Test Evaluation Activity

Tests

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

Test 1: The evaluator shall demonstrate that using a certificate without a valid certification path results in the selected action in the SFR. Using the administrative guidance, the evaluator shall then load a certificate or certificates to the Trust Anchor Database needed to validate the certificate to be used in the function, and demonstrate that the function succeeds. The evaluator then shall delete one of the certificates, and show that again, using a certificate without a valid certification path results in the selected action in the SFR.

This test was performed with conjunction of FIA_X509_EXT1.1 Test1. In this test, the evaluator first established a session with the TOE while having the CA installed in the TOE's trust store. Then the evaluator removed the CA from the trust store and attempted to connect to the TOE. The TOE terminated the session as it received an untrusted CA.

2.1.11 FCS_RBG_EXT.1 Random Bit Generation Services

2.1.11.1 TSS Evaluation Activity

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.

FCS_RBG_EXT.1.1 includes "implement DRBG functionality". As a result, the additional FCS_RBG_EXT.2 elements are included in the ST.

2.1.11.2 Guidance Evaluation Activity

None.

2.1.11.3 Test Evaluation Activity

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.

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

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.

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).

In FCS_RBG_EXT.1, the ST author has selected ***implement DRBG functionality***.

2.1.12 FCS_RBG_EXT.2 Random Bit Generation from Application

2.1.12.1 FCS_RBG_EXT.2.1

2.1.12.1.1 TSS Evaluation Activity

None.

2.1.12.1.2 Guidance Evaluation Activity

None.

2.1.12.1.3 Test Evaluation Activity

Tests

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 NIST-Recommended 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 – 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 – 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 implements CTR_DRBG (256 bits) according to NIST SP 800-90A and NIST SP 800-57. The correct operation of the algorithm was verified by CAVP Certificate A3617.

2.1.12.2 FCS_RBG_EXT.2.2

2.1.12.2.1 TSS Evaluation Activity

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

The Entropy Assessment Report was generated.

2.1.12.2.2 Guidance Evaluation Activity

None.

2.1.12.2.3 Test Evaluation Activity

Tests

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

2.1.13 FCS_STO_EXT.1 Storage of Credentials

2.1.13.1 TSS Evaluation Activity

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.

[ST] Section 6.2 states that the TOE uses OpenSSL to secure credential data at rest. Specifically, the TOE stores the following credentials:

- Web GUI authentication credentials: username and hashed password data for locally-defined users.
- Passphrases for certificate encryption: used to encrypt the TOE's TLS server certificate.

2.1.13.2 Guidance Evaluation Activity

None.

2.1.13.3 Test Evaluation Activity

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

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.

As stated in section 5.2.1.13 of the [ST] all credential storage is done by the TOE and not the platform. This requirement is covered by FCS_COP.1/SKC and FCS_CKM.1/PBKDF.

2.1.14 FCS_TLS_EXT.1 TLS Protocol (TLS Package)

2.1.14.1 TSS Evaluation Activity

None.

2.1.14.2 Guidance Evaluation Activity

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

In Section 5.2.1.14 of [ST] ("FCS_TLS_EXT.1 TLS Protocol"), "TLS as a server" is selected in FCS_TLS_EXT.1.1. The ST includes FCS_TLSS_EXT.1 from the selection-based requirements in [FPTLS], consistent with the selections made in FCS_TLS_EXT.1.1.

2.1.14.3 Test Evaluation Activity

None.

2.1.15 FCS_TLSS_EXT.1 TLS Server Protocol (TLS Package)

2.1.15.1 TSS Evaluation Activity

FCS_TLSS_EXT.1.1
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.

[ST] Section 6.2 identifies the supported cipher suites.

The TLS server implementation support the following TLS cipher suites in the TOE's evaluated configuration:

- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384

The cipher suites identified in the TSS correspond to those identified in the SFR.

FCS_TLSS_EXT.1.2
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.

[ST] Section 6.2 states that the TOE uses TLS 1.2 for server communications. In the case where the TOE acts as a TLS server, all other TLS versions are rejected.

FCS_TLSS_EXT.1.3
The evaluator shall verify that the TSS describes the key agreement parameters of the server's Key Exchange message.

[ST] Section 6.2 states that all supported ciphersuites use elliptic curves as the method of key establishment. The TSF presents secp256r1, secp384r1 and secp521r1 as the supported values in the Supported Groups extension and uses the same NIST curves for key establishment.

2.1.15.2 Guidance Evaluation Activity

FCS_TLSS_EXT.1.1
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.

The guidance provided by [User] includes instructions to the administrator on how to configure the TOE to use TLS conformant to the description in the TSS. Refer to "Additional Resources > Configure NNM for NIAP Compliance". Doing this will limit the TOE to using TLSS ciphers selected in the [ST].

FCS_TLSS_EXT.1.2
The evaluator shall verify that the AGD guidance includes any configuration necessary to meet this requirement.

The guidance provided by [User] includes instructions to the administrator on how to configure the TOE to use TLS conformant to the description in the TSS. Refer to "Additional Resources > Configure NNM for NIAP Compliance". Doing this will limit the TOE to using TLS 1.2.

FCS_TLSS_EXT.1.3
The evaluator shall verify that any configuration guidance necessary to meet the requirement must be contained in the AGD guidance.

The guidance provided by [User] includes instructions to the administrator on how to configure the TOE to use TLS conformant to the description in the TSS. Refer to "Additional Resources > Configure NNM for NIAP Compliance".

2.1.15.3 Test Evaluation Activity

FCS_TLSS_EXT.1.1
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).

The evaluator established a connection from a TLS test client to the TOE using each of the cipher suites claimed by the TOE. A wire capture was made of each connection to verify that the connection was completed using the claimed cipher suite.

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.

The evaluator attempted to establish a connection to the TOE with a TLS test client configured to use the TLS_CHACHA20_POLY1305_SHA256 cipher. The TOE terminated the connection attempt after receiving the Client Hello message. A second connection attempt was made with the TLS_NULL_WITH_NULL_NULL cipher. The TOE again terminated the connection attempt after receiving the Client Hello message.

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.

N/A. The TOE does not use RSA key exchange.

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

Modified per TD0469

Test 4.1: ~~Change the TLS version proposed by the client in the Client Hello to a non-supported TLS version (for example 1.3 represented by the two bytes 03 04) and verify that the server rejects the connection.~~

TD0469: Test 4.1 for FCS_TLSS_EXT.1.1 is removed from TLS package 1.1.

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.

The evaluator attempted to establish a connection to the TOE from a TLS test client that sent a modified Client Finished handshake message. The TOE terminated the connection after receiving the Client Finished message from the test client.

Modified per TD0588

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): Generate a Fatal Alert by sending a Finished message from the client before the client sends a ChangeCipherSpec message, and then send a Client Hello with the session identifier from the previous incomplete session, and verify that the server does not resume the session.

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.

The evaluator attempted to establish a connection to the TOE with a TLS test client that was configured to reuse a session ID from a previously terminated connection. The TOE terminated the connection attempt after receiving the reused session ID.

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.

The evaluator attempted to establish a connection to the TOE from a TLS test client that was configured to send garbled data after the client sent its ChangeCipherSpec message. The TOE terminated the connection after receiving the garbled data from the test client.

FCS_TLSS_EXT.1.2

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 used a TLS test client to attempt to open SSL 2.0 and 3.0 and TLS 1.0 and 1.1 connections to the TOE. All of these connection attempts were rejected by the TOE after it received the Client Hello messages.

FCS_TLSS_EXT.1.3

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.

N/A. The TOE does not use 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.

N/A. The TOE does not use finite-field Diffie-Hellman ciphers.

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.

The evaluator used a TLS test client to establish a TLS connection to the TOE using both of the curves claimed by the TOE.

2.1.16 FCS_TLSS_EXT.2 TLS Server Support for Mutual Authentication (TLS Package)

2.1.16.1 TSS Evaluation Activity

FCS_TLSS_EXT.2.1 / FCS_TLSS_EXT.2.2
The evaluator shall ensure that the TSS description required per FIA_X509_EXT.2.1 includes the use of client-side certificates for TLS mutual authentication.

[ST] Section 6.2 states that as part of certificate validation in the establishment of TLS connectivity, the TOE will validate the reference identifier of a presented server certificate. For all TOE usage of mutual TLS authentication, the TSF will perform the same verification of a presented client certificate. This is done through validation of the Common Name (CN) and Subject Alternative Name (SAN) certificate fields, the latter of which is expected to contain the FQDN of the external system that is presenting the certificate to the TOE. The reference identifier is established by configuration.

FCS_TLSS_EXT.2.3
If the product implements mutual authentication, the evaluator shall verify that the TSS describes how the DN and SAN in the certificate is compared to the expected identifier.

[ST] Section 6.2 states that the TOE supports mutual authentication. For all TOE usage of mutual TLS authentication, the TSF will perform the same verification of a presented client certificate. This is done through validation of the Common Name (CN) and Subject Alternative Name (SAN) certificate fields, the latter of which is expected to contain the FQDN of the external system that is presenting the certificate to the TOE. The reference identifier is established by configuration.

2.1.16.2 Guidance Evaluation Activity

FCS_TLSS_EXT.2.1 / FCS_TLSS_EXT.2.2

The evaluator shall verify that the AGD guidance required per FIA_X509_EXT.2.1 includes instructions for configuring the client-side certificates for TLS mutual authentication. The evaluator shall ensure that the AGD guidance includes instructions for configuring the server to require mutual authentication of clients using these certificates.

The guidance provided by [User] includes a description of how to configure the Nessus Network Monitor component to require mutual authentication of TLS clients. Refer to “Additional Resources > Configure Tenable Nessus Network Monitor for Certificates”.

Validation of administrator TLS client certificate

The guidance provided by [User] includes a description of how to create administrator TLS client certificates. Refer to “Additional Resources > Configure Tenable Nessus Network Monitor for Certificates > Create Tenable Nessus Network Monitor SSL Certificates for Login”.

Validation of Tenable.sc TLS client certificate

The guidance provided by [User] describes how the TOE can operate in conjunction with an instance of Tenable.sc in its operational environment. Refer to “Additional Resources > Working with Tenable.sc”. The Section references the Tenable.sc User Guide [TUG]. This document is not included in the evaluation but is referenced as a component in the operational environment.

The guidance provided by the Tenable.sc user guidance includes a description of how to add an instance of the TOE to Tenable.sc and configure TLS communications between Tenable.sc (the TLS client) and the TOE (the TLS server). The guidance also describes how to configure the client-side certificate used by Tenable.sc to authenticate itself to the TOE. Refer to “Configure Scans > Resources > Nessus Network Monitor Instances > Add an NNM Instance” in the Tenable.sc user guide.

FCS_TLSS_EXT.2.3
If the DN is not compared automatically to the domain name, IP address, username, or email address, the evaluator shall ensure that the AGD guidance includes configuration of the expected identifier or the directory server for the connection.

The guidance provided by [User] includes a description of how to create administrator TLS client certificates. Refer to “Additional Resources > Configure Tenable Nessus Network Monitor for Certificates > Create Tenable Nessus Network Monitor SSL Certificates for Login”.

2.1.16.3 Test Evaluation Activity

FCS_TLSS_EXT.2.1 / FCS_TLSS_EXT.2.2

The evaluator shall use TLS as a function to verify that the validation rules in FIA_X509_EXT.1.1 are adhered to and shall perform the following tests. The evaluator shall apply the AGD guidance to configure the server to require TLS mutual authentication of clients for the following tests, unless overridden by instructions in the test activity:

Test 1: The evaluator shall configure the server to send a certificate request to the client. The client shall send a certificate_list structure which has a length of zero. The evaluator shall verify that the handshake is not finished successfully and no application data flows.

The evaluator configured the TOE for mutual authentication. A connection attempt was then made from a TLS test client which responded to the TOE’s Certificate Request with a certificate_list structure of length zero. The TOE terminated the connection attempt after receiving the certificate from the client.

Test 2: The evaluator shall configure the server to send a certificate request to the client. The client shall send no client certificate message, and instead send a client key exchange message in an attempt to continue the handshake. The evaluator shall verify that the handshake is not finished successfully and no application data flows.

The evaluator configured the TOE for mutual authentication. A connection attempt was then made from a TLS test client which did not present a certificate in response to the TOE's Certificate Request message. The TOE terminated the connection after receiving the Client Key Exchange method.

Test 3: The evaluator shall configure the server to send a certificate request to the client without the supported_signature_algorithm used by the client's certificate.

The evaluator shall attempt a connection using the client certificate and verify that the handshake is not finished successfully and no application data flows.

The evaluator configured the TOE for mutual authentication. A connection attempt was then made from a TLS test client which sent a certificate which used the MD5 hash algorithm, which is not supported by the TOE. The TOE terminated the connection attempt after receiving the client's certificate.

Test 4: The evaluator shall demonstrate that using a certificate without a valid certification path results in the function failing. Using the administrative guidance, the evaluator shall then load a certificate or certificates needed to validate the certificate to be used in the function, and demonstrate that the function succeeds. The evaluator then shall delete one of the certificates, and show that the function fails.

The evaluator configured the TOE for mutual authentication. A connection attempt was then made from a TLS test client which sent a certificate issued by an intermediate CA. The TOE did not have the intermediate CA in its trust store, but it did have the root CA for the intermediate CA. The connection attempt succeeded when the test client sent its leaf certificate as well as the intermediate CA certificate. The connection failed when the TOE only sent the leaf certificate without the intermediate CA certificate.

Test 5: The aim of this test is to check the response of the server when it receives a client identity certificate that is signed by an impostor CA (either Root CA or intermediate CA). To carry out this test the evaluator shall configure the client to send a client identity certificate with an issuer field that identifies a CA recognized by the TOE as a trusted CA, but where the key used for the signature on the client certificate does not in fact correspond to the CA certificate trusted by the TOE (meaning that the client certificate is invalid because its certification path does not in fact terminate in the claimed CA certificate). The evaluator shall verify that the attempted connection is denied.

The evaluator configured the TOE for mutual authentication. A connection attempt was then made from a TLS test client which sent a certificate issued by a CA whose identifier was trusted by the TOE. The key used to sign the client certificate was change so that it did not correspond to the CA trusted by the TOE. The TOE terminated the connection after receiving the client's certificate.

Test 6: The evaluator shall configure the client to send a certificate with the Client Authentication purpose in the extendedKeyUsage field and verify that the server accepts the attempted connection. The evaluator shall repeat this test without the Client Authentication purpose and shall verify that the server denies the connection. Ideally, the two certificates should be identical except for the Client Authentication purpose.

The evaluator configured the TOE for mutual authentication. Two connection attempts were then made from a TLS test client to the TOE. The first client was configured to send a valid certificate while the second was configured to send a similar certificate that had Server Authentication in its extendedKeyUsage field rather than Client Authentication. The TOE accepted the valid certificate but rejected the connection attempt with the second certificate.

Test 7: The evaluator shall perform the following modifications to the traffic: a) Configure the server to require mutual authentication and then modify a byte in the client's certificate. The evaluator shall verify that the server rejects the connection. b) Configure the server to require mutual authentication and then

modify a byte in the signature block of the client's Certificate Verify handshake message. The evaluator shall verify that the server rejects the connection.

The evaluator configured the TOE for mutual authentication. Two connection attempts were then made from a TLS test client to the TOE. The first attempt transmitted a modified client certificate and the second attempt modified a byte in the signature block of the client's Certificate Verify message. The TOE rejected both of these connection attempt.

FCS_TLSS_EXT.2.3
Test 1: The evaluator shall send a client certificate with an identifier that does not match any of the expected identifiers and verify that the server denies the connection. The matching itself might be performed outside the TOE (e.g. when passing the certificate on to a directory server for comparison).

The evaluator configured the TOE for mutual authentication. A connection attempt was established from a TLS test client that was configured to send a certificate with an invalid reference identifier. The TOE terminated the connection attempt after receiving the client certificate.

2.2 User Data Protection (FDP)

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

2.2.1.1 TSS Evaluation Activity

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.

[ST] Section 6.3 identifies the sensitive data processed and stored by the TOE:

Sensitive Data	Exchange	Protection at Rest
GUI credentials	Admin's browser to Web Server over browser connection	FCS_STO_EXT.1 (PBKDF)
Passphrase for PKI certificate encryption	None	FCS_STO_EXT.1 (AES)
Collected network traffic data	NNM to Tenable.sc	FCS_STO_EXT.1 (AES)

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.

Not applicable. Sensitive data at rest is protected by the TOE's implementation of AES.

2.2.1.2 Guidance Evaluation Activity

None.

2.2.1.3 Test Evaluation Activity

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: 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: 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.

The guidance provided by [User] makes clear to the administrator the need to activate platform encryption in the evaluated configuration.

2.2.2 FDP_DAR_EXT.1(2) Encryption Of Sensitive Application Data

2.2.2.1 TSS Evaluation Activity

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.

[ST] Section 6.3 identifies the sensitive data processed and stored by the TOE:

Sensitive Data	Exchange	Protection at Rest
Database encryption key (used for all AES operations)	None	FDP_DAR_EXT.1(2)

The database encryption key is generated by the TOE's DRBG as part of the initial setup process. The key is stored as a read-only file owned by root or SYSTEM, depending on platform. The administrator has the ability to optionally set a passphrase to unlock the use of this key; if this option is chosen, they must enter the passphrase when the TOE first starts. The database encryption key is protected cryptographically by the platform's use of full disk encryption.

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.

Not applicable. The AES key used by the TOE to protect sensitive data at rest is protected in turn by the platform's use of full disk encryption.

2.2.2.2 Guidance Evaluation Activity

None.

2.2.2.3 Test Evaluation Activity

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: 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: 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.

The evaluator displayed the contents of where the user credentials are stored and noticed that the file is unreadable.

2.2.3 FDP_DEC_EXT.1 Access to Platform Resources

2.2.3.1 FDP_DEC_EXT.1.1

2.2.3.1.1 TSS Evaluation Activity

None.

2.2.3.1.2 Guidance Evaluation Activity

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 6.3 of [ST] (“User Data Protection”) states the TOE uses network connectivity for remote management, retrieval of scan results by Tenable.sc, retrieval of plugin updates from Tenable.sc, and collection of network data.

The guidance provided by [User] includes the following descriptions of the TOE’s access to network resources:

- For remote management, refer to “Welcome to Nessus Network Monitor > Set up Tenable Nessus Network Monitor > Configure Tenable Nessus Network Monitor”, which describes the use of a web browser to connect to the TOE in order to configure it.
- For retrieval of scan results by Tenable.sc, refer to “Additional Resources > Working with Tenable.sc”.

- For retrieval of plugin updates from Tenable.sc, refer to “Configuration Page > Feed Settings Section > Download New Vulnerability Plugins”.
- For collection of network data, refer to [USER] “Welcome to Nessus Network Monitor > Set up Tenable Nessus Network Monitor > Configure Tenable Nessus Network Monitor”.

2.2.3.1.3 Test Evaluation Activity

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

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.

This test is satisfied by documentation provided with the TOE and does not require any testing activity.

[User] Section “Welcome to Tenable Nessus Network Monitor > System Requirements > Tenable Nessus Network Monitor Hardware Requirements” describes the hardware use of the TOE.

2.2.3.2 FDP_DEC_EXT.1.2

2.2.3.2.1 TSS Evaluation Activity

None.

2.2.3.2.2 Guidance Evaluation Activity

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.

The statement of FDP_DEC_EXT.1.2 in Section 5.2.2.3 of [ST] (“FDP_DEC_EXT.1 Access to Platform Resources”) specifies “system logs” as the only sensitive information repository accessed by the TOE.

The guidance provided by [User] includes a description of how the TOE can be configured to write its own debug messages to the platform’s system logs. Refer to “Configuration Page > Tenable Nessus Network Monitor Settings Section”, > the subsection “Tenable Nessus Network Monitor Web Server” subsection “Enable Debug Logging for Tenable Nessus Network Monitor Web Server”.

2.2.3.2.3 Test Evaluation Activity

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

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.

This test is satisfied by documentation provided with the TOE and does not require any testing activity.

Section 5.2.2.3 of the [ST] states that the TOE restricts its access to system logs and system configuration. Section 6.3 clarifies that the TOE accesses system configuration to collect data about the local system for subsequent analysis, and accesses system logs on the local system (/var/log/messages or Windows Event Log) to record data about its own behavior.

2.2.4 FDP_NET_EXT.1 Network Communications

2.2.4.1 TSS Evaluation Activity

None.

2.2.4.2 Guidance Evaluation Activity

None.

2.2.4.3 Test Evaluation Activity

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.

The evaluator ran a wire capture on the TOE's platform while the TOE was running. The traffic was examined and all non-user initiated traffic was standard management traffic for the RHEL and Windows Server platform.

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).

An NMAP port scan was performed on the TOE. The only open ports on the RHEL instance of the TOE were those used for SSH and TLS/HTTPS access. The Windows instance of the TOE only opened a port for TLS/HTTPS access. All other open ports were standard management services for the Windows Server platform.

2.3 Identification and Authentication (FIA)

2.3.1 FIA_X509_EXT.1 X.509 Certificate Validation

2.3.1.1 FIA_X509_EXT.1.1

2.3.1.1.1 TSS Evaluation Activity

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

[ST] Section 6.4 states that the TOE's use of the certificate validation function is to validate the authenticity of remote endpoints, the TSF chooses what certificates to use based on what is presented to it as part of establishing the TLS session. The TOE is only assigned one certificate for its own use, so there is only one certificate that it will present in cases where a remote entity may need to validate it.

The TSS describes the certificate path validation algorithm as defined in RFC 5280.

2.3.1.1.2 Guidance Evaluation Activity

None.

2.3.1.1.3 Test Evaluation Activity

The tests described must be performed in conjunction with the other certificate services evaluation activities, including the functions in FIA_X509_EXT.2.1. The tests for the extendedKeyUsage rules are performed in conjunction with the uses that require those rules. If the application supports chains of length four or greater, the evaluator shall create a chain of at least four certificates: the node certificate to be tested, two Intermediate CAs, and the self-signed Root CA. If the application supports a maximum trust depth of two, then a chain with no Intermediate CA should instead be created.

Test 1: The evaluator shall demonstrate that validating a certificate without a valid certification path results in the function failing, for each of the following reasons, in turn:

- by establishing a certificate path in which one of the issuing certificates is not a CA certificate,
- by omitting the basicConstraints field in one of the issuing certificates,
- by setting the basicConstraints field in an issuing certificate to have CA=False,
- by omitting the CA signing bit of the key usage field in an issuing certificate, and
- by setting the path length field of a valid CA field to a value strictly less than the certificate path.

The evaluator shall then establish a valid certificate path consisting of valid CA certificates, and demonstrate that the function succeeds. The evaluator shall then remove trust in one of the CA certificates, and show that the function fails.

- The evaluator used a client certificate with and multi-level trust chain to connect to the TOE. Because the entire chain was present, the TOE was able to authenticate the client and establish a successful TLS session. Next the evaluator removed the top intermediate certificate that tied the rest of the certificate to the root CA which is configured on the TOE. Using the incomplete certificate chain, the evaluator attempted to connect to the TOE, but the TOE terminated the session.
- The TOE attempted to connect to the TOE with a TLS test client whose leaf certificate was issued by an intermediate CA that did not have the basicConstraints extension. The TOE terminated the connection attempt after receiving the client certificate.

- TOE attempted to connect to the TOE with a TLS test client whose leaf certificate was issued by an intermediate CA that had the basicConstraints extension set to FALSE. The TOE terminated the connection attempt after receiving the client certificate.
- The evaluator attempted to connect to the TOE while only presenting a certificate and an intermediate certificate authority which had its Certificate Signing bit removed from it's KeyUsage field. The TOE terminated the session due to the invalid CA.
- The evaluator configured a TLS client to present a leaf certificate and an issuer that had a pathlength of zero and proceeded to connect to the TOE. The TOE properly completed the connection successfully. Next the evaluator configured the same client with a sperate leaf certificate and intermediate CA that was created from the previous (pathlength zero) intermediate CA. The evaluator then attempted to perform a TLS connection to the TOE and witnessed that the TOE terminated the session.

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

The evaluator attempted to open a TLS connection from a TLS test client to the TOE. The test client was configured to identify itself with a certificate that had expired. The TOE terminated the connection attempt after receiving the client's certificate.

Test 3: The evaluator shall test that the TOE can properly handle revoked certificates-–conditional on whether CRL, OCSP, OCSP Stapling or OCSP Multi-stapling is selected; if multiple methods are selected, then the following tests shall be performed for each method:

- The evaluator shall test revocation of the node certificate.
- The evaluator shall also test revocation of an intermediate CA certificate (i.e. the intermediate CA certificate should be revoked by the root CA), if intermediate CA certificates are supported. If OCSP Stapling per RFC 6066 is the only supported revocation method, this test is omitted.
- The evaluator shall ensure that a valid certificate is used, and that the validation function succeeds. The evaluator then attempts the test with a certificate that has been revoked (for each method chosen in the selection) to ensure when the certificate is no longer valid that the validation function fails.

The evaluator attempted to open a TLS connection from a TLS test client to the TOE. The test client identified itself with a certificate configured for OCSP. The TOE was able to successfully establish a connection when the certificate. When OCSP was used to revoke the test client certificate the TOE terminated the connection attempt.

Modified in accordance with TD0669.

Test 4: If any OCSP option is selected, the evaluator shall ensure the TSF has no other source of revocation information available and configure the OCSP server or use a man-in-the-middle tool to present an OCSP response signed by a certificate that does not have the OCSP signing purpose and which is the only source of revocation status information advertised by the CA issuing the certificate being validated. The evaluator shall verify that validation of the OCSP response fails and that the TOE treats the certificate being checked as invalid and rejects the connection. If CRL is selected, the evaluator shall likewise configure the CA to be the only source of revocation status information, and sign a CRL with a certificate that does not have the cRLsign key usage bit set~~, and~~. The evaluator shall verify that validation of the CRL fails and that the TOE treats the certificate being checked as invalid and rejects the connection.

Note: The intent of this test is to ensure a TSF does not trust invalid revocation status information. A TSF receiving invalid revocation status information from the only advertised certificate status provider should treat the certificate whose status is being checked as invalid. This should generally be treated differently from the case where the TSF is not able to establish a connection to check revocation status information, but it is acceptable that the TSF ignore any invalid information and attempt to find another source of revocation status (another advertised provider, a locally configured provider, or cached information) and treat this situation as not having a connection to a valid certificate status provider.

The evaluator attempted to open a connection from a TLS test client to the TOE. The test client identified itself with a certificate configured for OCSP. The OCSP responder's certificate was modified so that it did not have the OCSP signing purpose. The TOE terminated the TLS session as soon as it received that invalid OCSP signer.

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

The evaluator attempted to open a connection from a TLS test client to the TOE. The test client's certificate had one of its first eight bytes modified. The TOE terminated the connection attempt after receiving the client certificate.

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

The evaluator attempted to open a connection from a TLS test client to the TOE. The test client's certificate had one of its last eight bytes modified. The TOE terminated the connection attempt after receiving the client certificate.

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

The evaluator attempted to open a connection from a TLS test client to the TOE. The test client's certificate had its public key modified. The TOE terminated the connection attempt after receiving the client certificate.

Test 8: (Conditional on support for EC certificates as indicated in FCS_COP.1/Sig). The evaluator shall establish a valid, trusted certificate chain consisting of an EC leaf certificate, an EC Intermediate CA certificate not designated as a trust anchor, and an EC certificate designated as a trusted anchor, where the elliptic curve parameters are specified as a named curve. The evaluator shall confirm that the TOE validates the certificate chain.

N/A, the TOE only supports RSA certificates.

Test 9: (Conditional on support for EC certificates as indicated in FCS_COP.1/Sig). The evaluator shall replace the intermediate certificate in the certificate chain for Test 8a with a modified certificate, where the modified intermediate CA has a public key information field where the EC parameters uses an explicit format version of the Elliptic Curve parameters in the public key information field of the intermediate CA certificate from Test 8a, and the modified Intermediate CA certificate is signed by the trusted EC root CA, but having no other changes. The evaluator shall confirm the TOE treats the certificate as invalid.

N/A, the TOE only supports RSA certificates.

2.3.1.2 FIA_X509_EXT.1.2

2.3.1.2.1 TSS Evaluation Activity

None.

2.3.1.2.2 Guidance Evaluation Activity

None.

2.3.1.2.3 Test Evaluation Activity

The tests described must be performed in conjunction with the other certificate services evaluation activities, including the functions in FIA_X509_EXT.2.1. If the application supports chains of length four or greater, the evaluator shall create a chain of at least four certificates: the node certificate to be tested, two Intermediate CAs, and the self-signed Root CA. If the application supports a maximum trust depth of two, then a chain with no Intermediate CA should instead be created.

Test 1: The evaluator shall ensure that the certificate of at least one of the CAs in the chain does not contain the basicConstraints extension. The evaluator shall confirm that validation of the certificate path fails (i) as part of the validation of the peer certificate belonging to this chain; and/or (ii) when attempting to add the CA certificate without the basicConstraints extension to the TOE's trust store.

The following test has been performed with conjunction of FIA_X509_EXT1.1 Test 1. Within the test perform for FIA_X509_EXT.1.1 Test 1, one of the tests required the TOE to terminate a TLS handshake while the CA had an invalid basicConstraint that was NOT set. The TOE properly terminated the session as it received the client certificate.

Test 2: The evaluator shall ensure that the certificate of at least one of the CAs in the chain has the CA flag in the basicConstraints extension not set (or set to FALSE). The evaluator shall confirm that validation of the certificate path fails (i) as part of the validation of the peer certificate belonging to this chain; and/or (ii) when attempting to add the CA certificate with the CA flag not set (or set to FALSE) in the basicConstraints extension to the TOE's trust store.

The following test has been performed with conjunction of FIA_X509_EXT1.1 Test 1. Within the test perform for FIA_X509_EXT.1.1 Test 1, one of the tests required the TOE to terminate a TLS handshake while the CA had an invalid basicConstraint set to FALSE. The TOE properly terminated the session as it received the client certificate.

2.3.2 FIA_X509_EXT.2 X.509 Certificate Authentication

2.3.2.1 TSS Evaluation Activity

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

[ST] Section 6.4 states that the TOE's use of the certificate validation function is to validate the authenticity of remote endpoints, the TSF chooses what certificates to use based on what is presented to it as part of establishing the TLS session. The TOE is only assigned one certificate for its own use, so there is only one certificate that it will present in cases where a remote entity may need to validate it.

The evaluator shall examine the TSS to confirm that it describes the behavior of the TOE when a connection cannot be established during the validity check of a certificate used in establishing a trusted channel. The evaluator shall verify that any distinctions between trusted channels are described.

[ST] Section 6.4 states that if the revocation status of a certificate cannot be verified (i.e. the OCSP responder cannot be reached), the TOE will accept the certificate.

If the requirement that the administrator is able to specify the default action, then the evaluator shall ensure that the operational guidance contains instructions on how this configuration action is performed.

The administrator is not required to configure the default action. The TOE will accept the certificate if the OCSP responder cannot be reached.

2.3.2.2 Guidance Evaluation Activity

None.

2.3.2.3 Test Evaluation Activity

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

Test 1: The evaluator shall demonstrate that using a valid certificate that requires certificate validation checking to be performed in at least some part by communicating with a non-TOE IT entity. The evaluator shall then manipulate the environment so that the TOE is unable to verify the validity of the certificate, and observe that the action selected in FIA_X509_EXT.2.2 is performed. If the selected action is administrator-configurable, then the evaluator shall follow the operational guidance to determine that all supported administrator-configurable options behave in their documented manner.

This test was covered by FIA_X509_EXT.1.1 Test 3. That test required the TOE to perform certificate validation by connecting to a non-TOE IT entity (in this case an OCSP responder). Next the evaluator disabled one of the OCSP responder so that the TOE would not be able to validate the OCSP certificate. The TOE was able to complete the TLS session even though it was not able to get the status of the certificate.

Test 2: The evaluator shall demonstrate that an invalid certificate that requires certificate validation checking to be performed in at least some part by communicating with a non-TOE IT entity cannot be accepted.

This test is covered by testing of FIA_X509_EXT.1.1 Test 3 and Test 4. In those respective tests, the evaluator attempts to connect to the TOE while presenting a client certificate that had a unrevoked/revoked OCSP intermediate CA and entity certificate, and an OCSP certificate which had a OCSP signer bit missing from it's responder certificate.

2.4 Security Management (FMT)

2.4.1 FMT_CFG_EXT.1 Secure by Default Configuration

2.4.1.1 FMT_CFG_EXT.1.1

2.4.1.1.1 TSS Evaluation Activity

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

[ST] section 6.5 states that the TOE provides a web-based graphical user interface (GUI) that requires user authentication to access. The TOE has a default administrator credential of admin/admin that must be changed on first use. Administrator credentials are stored locally and protected by the TSF as per FCS_STO_EXT.1. Following the initial installation, additional accounts can be created.

2.4.1.1.2 Guidance Evaluation Activity

None.

2.4.1.1.3 Test Evaluation Activity

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.

It is not possible to run the TOE without credentials. As described in the [User] a username and password must be specified during the installation process.

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.

It is not possible to clear all credentials on the TOE. The evaluator demonstrated that an administrative account may not delete itself, so if the TOE is reduced to one account it cannot be deleted or cleared.

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.

The evaluator attempted to login to the TOE using the default credentials that were used in order finalize the installation process and create new credentials. The default credentials did not work to access the TOE.

2.4.1.2 FMT_CFG_EXT.1.2

2.4.1.2.1 TSS Evaluation Activity

None.

2.4.1.2.2 Guidance Evaluation Activity

None.

2.4.1.2.3 Test Evaluation Activity

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: 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: 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.

No world-writable files were found inside of the TOE's data directories.

2.4.2 FMT_MEC_EXT.1 Supported Configuration Mechanism

2.4.2.1 TSS Evaluation Activity

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.

[ST] Section 6.5 states that all directories containing TOE software and data are configured by default in such a manner that nothing is world-writable on Linux and Administrator privileges are required to access them on Windows. Configuration settings that affect the TOE's interaction with the host OS platform are stored in /etc for Linux and the Windows Registry for Windows.

The TOE supports the following security-relevant management and configuration settings:

- Configuration of transmission of system's hardware, software, or configuration information
    - Configuration of collection of network traffic data.

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.

Not Applicable. The SFR does not include the selection to "implement functionality to encrypt and store configuration options as defined by FDP_PRT_EXT.1 in the PP-Module for File Encryption".

2.4.2.2 Guidance Evaluation Activity

None.

2.4.2.3 Test Evaluation Activity

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:

Modified in accordance with TD0624.

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: 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:\ProgramData\ directory.

The evaluator modified the TOE's settings for certificate verification depth while monitoring the TOE with ProcMon (for the Windows platform). For the RHEL platform, evaluator took the hash of the configuration file before and after the modification of the TOE's setting. Changes were written to the appropriate directories for the application.

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 [ST] does not make this selection.

2.4.3 FMT_SMF.1 Specification of Management Functions

2.4.3.1 TSS Evaluation Activity

None.

2.4.3.2 Guidance Evaluation Activity

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.

As described in Section 6.5 of [ST] ("Security Management"), the TOE supports the following security-relevant management function: configuration of transmission of system's hardware, software, or configuration information—specifically, configuration of collection of network traffic data.

The guidance provided by [User] includes instructions to the administrator on how to configure collection of network traffic data. Refer to "Welcome to Nessus Network Monitor > Set up Tenable Nessus Network Monitor > Configure Tenable Nessus Network Monitor" for guidance on configuring collection of network traffic data.

2.4.3.3 Test Evaluation Activity

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 demonstrated the ability to access the management functionality specified in [User].

2.5 Privacy (FPR)

2.5.1 FPR_ANO_EXT.1 User Consent for Transmission of Personally Identifiable Information

2.5.1.1 TSS Evaluation Activity

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

[ST] Section 6.6 states that the TOE prevents the unnoticed/unauthorized transmission of PII across a network by not having functionality that is intended for such transmissions.

2.5.1.2 Guidance Evaluation Activity

None.

2.5.1.3 Test Evaluation Activity

If ***require user approval before executing*** is selected, the evaluator shall run the application and exercise the functionality responsible for transmitting PII and verify that user approval is required before transmission of the PII.

Section 6.6 of [ST] ("Privacy") states the TOE is not responsible for the collection of PII and does not transmit PII over a network.

2.6 Protection of the TSF (FPT)

2.6.1 FPT_AEX_EXT.1 Anti-Exploitation Capabilities

2.6.1.1 FPT_AEX_EXT.1.1

2.6.1.1.1 TSS Evaluation Activity

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

[ST] Section 6.7 states that the TOE implements address space layout randomization (ASLR) through the use of the /DYNAMICBASE (Windows) and –f PIC (Linux) compiler flags and relies fully on its underlying host platforms to perform memory mapping.

2.6.1.1.2 Guidance Evaluation Activity

None.

2.6.1.1.3 Test Evaluation Activity

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: 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: 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.

For the Linux TOE instance the evaluator ran the TOE on two different Linux systems. A comparison of their memory maps showed that the two instances did not share memory mapping locations.

For the Windows TOE instance the evaluator ran the TOE on two different Windows systems. A comparison of VMMap output from the two different systems showed that the two instances did not share memory mapping locations.

2.6.1.2 FPT_AEX_EXT.1.2

2.6.1.2.1 TSS Evaluation Activity

None.

2.6.1.2.2 Guidance Evaluation Activity

None.

2.6.1.2.3 Test Evaluation Activity

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

- `mmap` is never be invoked with both the `PROT_WRITE` and `PROT_EXEC` permissions, and
- `mprotect` is never invoked with the `PROT_EXEC` permission.

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.

Searches were performed through the application's source code for the strings "PROT_EXEC" and "PROT_WRITE". These strings were never used in the TOE's code.

2.6.1.3 FPT_AEX_EXT.1.3

2.6.1.3.1 TSS Evaluation Activity

None.

2.6.1.3.2 Guidance Evaluation Activity

None.

2.6.1.3.3 Test Evaluation Activity

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

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: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/threat-protection/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).

The evaluator verified that the Linux instance of the TOE could run on a system with SELinux enabled.

The evaluator used EMET to verify that none of the required protections were disabled on a Windows system running the TOE.

2.6.1.4 FPT_AEX_EXT.1.4

2.6.1.4.1 TSS Evaluation Activity

None.

2.6.1.4.2 Guidance Evaluation Activity

None.

2.6.1.4.3 Test Evaluation Activity

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: 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: 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.

For the Linux instance of the TOE the evaluator went into /opt/nnm (the installation directory specified in the [ST]) and its subdirectories and examined it for file permissions. All executable files were shown to be in different directories than those with read/write permissions.

For the Windows instance of the TOE this test was performed in conjunction with FMT_CFG_EXT.1.2. The access check tool used in that test demonstrated that the TOE's executable files were not in the same directories as any user modifiable files.

2.6.1.5 FPT_AEX_EXT.1.5

2.6.1.5.1 TSS Evaluation Activity

None.

2.6.1.5.2 Guidance Evaluation Activity

None.

2.6.1.5.3 Test Evaluation Activity

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.

There are no evaluation activities for this requirement for Linux based TOEs. The TOE does not have any PE or ELF executables.

For the Windows instance of the TOE the development environment was examined to verify that the /GS flag was used during compilation.

2.6.2 FPT_API_EXT.1 Use of Supported Services and APIs

2.6.2.1 TSS Evaluation Activity

The evaluator shall verify that the TSS lists the platform APIs used in the application.

[ST] Section 6.7 states that the TOE uses only documented platform APIs. Appendix A.1 identifies the platform APIs for both the Windows and Linux applications.

2.6.2.2 Guidance Evaluation Activity

None.

2.6.2.3 Test Evaluation Activity

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.

Appendix A.1 of [ST] ("Platform APIs") lists the platform APIs used by the TOE. All were found to be valid APIs.

2.6.3 FPT_LIB_EXT.1 Use of Third Party Libraries

2.6.3.1 TSS Evaluation Activity

None.

2.6.3.2 Guidance Evaluation Activity

None.

2.6.3.3 Test Evaluation Activity

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 surveyed the TOE's installation directory for dynamic libraries and found that none were present.

2.6.4 FPT_IDV_EXT.1 Software Identification and Versions

2.6.4.1 TSS Evaluation Activity

If **"other version information"** is selected the evaluator shall verify that the TSS contains an explanation of the versioning methodology.

"Other version information" is identified in the security requirement. [ST] Section 6.7 identifies the TOE versioning as using semver (Semantic Versioning) in the format x.y(.z) where x is the major version, y is the minor version, and the optional z is the patch version; SWID is not used.

2.6.4.2 Guidance Evaluation Activity

None.

2.6.4.3 Test Evaluation Activity

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.

As shown in FPT_TUD_EXT.1.1, the TOE's version is 6.2.2 .

2.6.5 FPT_TUD_EXT.1 Integrity for Installation and UpdateFPT_TUD_EXT.1.1

2.6.5.1.1 TSS Evaluation Activity

None.

2.6.5.1.2 Guidance Evaluation Activity

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

The guidance provided by [User] includes a description of how updates are performed on both Windows and Linux platforms. Refer to "Welcome to Nessus Network Monitor > Upgrade Tenable Nessus Network Monitor > Upgrade Tenable Nessus Network Monitor on Windows", and "Welcome to Nessus Network Monitor > Upgrade Tenable Nessus Network Monitor > Upgrade Tenable Nessus Network Monitor on Linux".

2.6.5.1.3 Test Evaluation Activity

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 verified that the TOE's Information menu contained information on the TOE's current version as well a link to Tenable's Help & Support page which provides any product updates.

2.6.5.2 FPT_TUD_EXT.1.2

2.6.5.2.1 TSS Evaluation Activity

None.

2.6.5.2.2 Guidance Evaluation Activity

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

Section 6.7 of [ST] ("Protection of the TSF") states the current version of the TOE can be queried in the following ways: invoking the TOE binary from the command line with the –v flag; via the web GUI; through Tenable.sc.

The guidance provided by [User] describes how to query the current version of the application as follows:

- by invoking the application binary with the –v option. Refer to [USER] "Additional Resources > Command Line Operations > Common Command Line Operations"
- via the web GUI. Refer to "Welcome to Nessus Network Monitor > View Tenable Nessus Network Monitor Information".

2.6.5.2.3 Test Evaluation Activity

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 verified that the TOE's Information menu contained information on the TOE's current version as well a link to Tenable's Help & Support page which provides any product updates.

2.6.5.3 FPT_TUD_EXT.1.3

2.6.5.3.1 TSS Evaluation Activity

None.

2.6.5.3.2 Guidance Evaluation Activity

None.

2.6.5.3.3 Test Evaluation Activity

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.

For the Linux instance of the TOE the evaluator obtained a hash of all of the files in the /opt/nnm/bin directory. After performing some testing activity a second hash of these files was generated. It was found to be unchanged from the first.

For the Windows instance of the TOE the evaluator obtained a hash of the nnm.exe executable. After performing some testing activity a second hash of the file was generated. It was found to be unchanged from the first.

2.6.5.4 FPT_TUD_EXT.1.4

2.6.5.4.1 TSS Evaluation Activity

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.

[ST] Section 6.7 states that updates the TOE are digitally signed by Tenable using 2048-bit RSA. The updates are validated by the host platform prior to installation.

The TOE can leverage its OS platform to check for software updates and acquire them if they are available. In this case, candidate updates are obtained by the administrator downloading them directly from Tenable's website or through a package manager such as yum.

2.6.5.4.2 Guidance Evaluation Activity

None.

2.6.5.4.3 Test Evaluation Activity

None.

2.6.5.5 FPT_TUD_EXT.1.5

2.6.5.5.1 TSS Evaluation Activity

The evaluator shall verify that the TSS identifies how the application is distributed. 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 is distributed as an additional software package to the platform OS.

The TOE will not download, modify, replace, or update its own binary code. The TOE is packaged as an .rpm file for Linux and an .exe file for Windows. Each are digitally signed by Tenable using 2048-bit RSA. Removing (uninstalling) the product will remove all executable code from the host system.

2.6.5.5.2 Guidance Evaluation Activity

None.

2.6.5.5.3 Test Evaluation Activity

None.

2.6.6 FPT_TUD_EXT.2 Integrity for Installation and Update

2.6.6.1 FPT_TUD_EXT.2.1

2.6.6.1.1 TSS Evaluation Activity

None.

2.6.6.1.2 Guidance Evaluation Activity

None.

2.6.6.1.3 Test Evaluation Activity

Modified in accordance with TD0628.

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: 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: 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.

The evaluator verified that the installation file for the Linux instance of the TOE is packaged as a .rpm file.

The evaluator verified that the installation file for the Windows instance of the TOE is a .exe file.

2.6.6.2 FPT_TUD_EXT.2.2

2.6.6.2.1 TSS Evaluation Activity

None.

2.6.6.2.2 Guidance Evaluation Activity

None.

2.6.6.2.3 Test Evaluation Activity

Modified in accordance with TD0664.

~~**Platforms: Microsoft Windows...**~~

~~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.~~

~~**Platforms: Linux**~~

~~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.~~

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.

Both instances of the TOE were uninstalled following instructions provided in [TUG].

Upon examination it was found that for the Linux TOE platform the /opt/nnm directory that the TOE had been installed into was no longer present and all of the TOE's files had been removed from the filesystem.

It was found that for the Windows TOE platform the Program Files/Tenable/NNM directory that the TOE had been installed into was no longer present and all of the TOE's files had been removed from the filesystem.

2.6.6.3 FPT_TUD_EXT.2.3

2.6.6.3.1 TSS Evaluation Activity

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.

[ST] Section 6.7 states that updates the TOE are digitally signed by Tenable using 2048-bit RSA. The updates are validated by the host platform prior to installation.

2.6.6.3.2 Guidance Evaluation Activity

None.

2.6.6.3.3 Test Evaluation Activity

None.

2.7 Trusted Path/Channels (FTP)

2.7.1 FTP_DIT_EXT.1 Protection of Data in Transit

2.7.1.1 TSS Evaluation Activity

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

The TOE does not rely on platform-provided functionality to encrypt transmitted sensitive data. [ST] Section 6.8 states that in the evaluated configuration, the TOE uses its own cryptographic implementation to encrypt sensitive data in transit.

2.7.1.2 Guidance Evaluation Activity

None.

2.7.1.3 Test Evaluation Activity

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.

These tests were performed in conjunction with the tests for FCS_TLSS_EXT.1. As those activities showed all data transmitted to and from the TOE was protected by TLS 1.2. No unencrypted traffic or plaintext data was sent.

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.

These tests were performed in conjunction with the tests for FCS_TLSS_EXT.1. As those activities showed all data transmitted to and from the TOE was protected by TLS 1.2. No unencrypted traffic or plaintext data was sent.

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.

These tests were performed in conjunction with the tests for FCS_TLSS_EXT.1. As those activities showed all data transmitted to and from the TOE was protected by TLS 1.2. No unencrypted traffic or plaintext data was sent.

3. Security Assurance Requirement Assurance Activities

3.1 Development (ADV)

3.1.1 Basic Functional Specification (ADV_FSP.1)

3.1.1.1 Assurance Activity

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 identified above provided sufficient information to determine the appropriate content for the TSS section and to perform the assurance activities. Since these are directly associated with the SFRs, and are implicitly already done, no additional documentation or analysis is necessary.

3.2 Guidance Documents (AGD)

3.2.1 Operational User Guidance (AGD_OPE.1)

3.2.1.1 Assurance Activity

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.

The guidance provided by [User] includes a description of how updates are performed on both Windows and Linux platforms. Refer to “Welcome to Tenable Nessus Network Monitor > “Upgrade Tenable Nessus Network Monitor” > “Upgrade Tenable Nessus Network Monitor on Linux” and “Welcome to Tenable Nessus Network Monitor > “Upgrade Tenable Nessus Network Monitor” > Upgrade Tenable Nessus Network Monitor on Windows”.

The guidance provided by [User] includes instructions to the administrator on how to configure the TOE so that it uses the cryptographic functions described in the Security Target. Refer to "Additional Resources > Configure NNM for NIAP Compliance".

3.2.2 Preparative Procedures (AGD_PRE.1)

3.2.2.1 Assurance Activity

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 TOE in its evaluated configuration is supported on RHEL 8.7 and Windows Server 2019 platforms that are adequately addressed in the guidance documentation. "Welcome to Tenable Nessus Network Monitor > System Requirements > Tenable Nessus Network Monitor Hardware Requirements" and "Welcome to Tenable Nessus Network Monitor > System Requirements > Tenable Nessus Network Monitor Software Requirements" identifies the operational environment needed to support the TOE.

3.3 Tests (ATE)

3.3.1 Independent Testing – Conformance (ATE_IND.1)

3.3.1.1 Assurance Activity

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.

Testing of the TOE was performed at the Leidos Accredited Testing and Evaluation Lab located in Columbia, Maryland from October 2022 to June 2023.

3.4 Vulnerability Assessment (AVA)

3.4.1 Vulnerability Survey (AVA_VAN.1)

3.4.1.1 Assurance Activity

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.

For Windows, Linux, macOS and Solaris: 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 evaluation team performed a search of the following public vulnerability database:

- National Vulnerability Database (https://nvd.nist.gov/)
- Tenable CVEs: (https://www.tenable.com/cve)
- OpenSSL Vulnerabilities: (https://www.openssl.org/news/vulnerabilities-3.0.html)
- Carnegie Mellon University CERT Coordination Center https://www.kb.cert.org/vuls/search/

Searches were performed on 27 April 2023. An updated search was performed on 22 May 2023. A final search was performed on 21 June 2023.

Note that Tenable, as a vulnerability research and management company, maintains its own repository of publicly known vulnerabilities. As such, any search hits that are due solely to Tenable being a reference for the vulnerability announcement or security advisory are excluded from the presented search results.

The following search terms were used in the in the searches of the repositories listed above:

- "tenable"

- “nessus” (Note, as discussed in [ST], “Nessus Manager” is the same product as “Nessus”, with an additional license enabled, so searching on “nessus” encompasses searching for “nessus manager”)
- “Network Monitor”
- “tls v1.2”
- “openssl 3.0.9”
- Third-Party Libraries

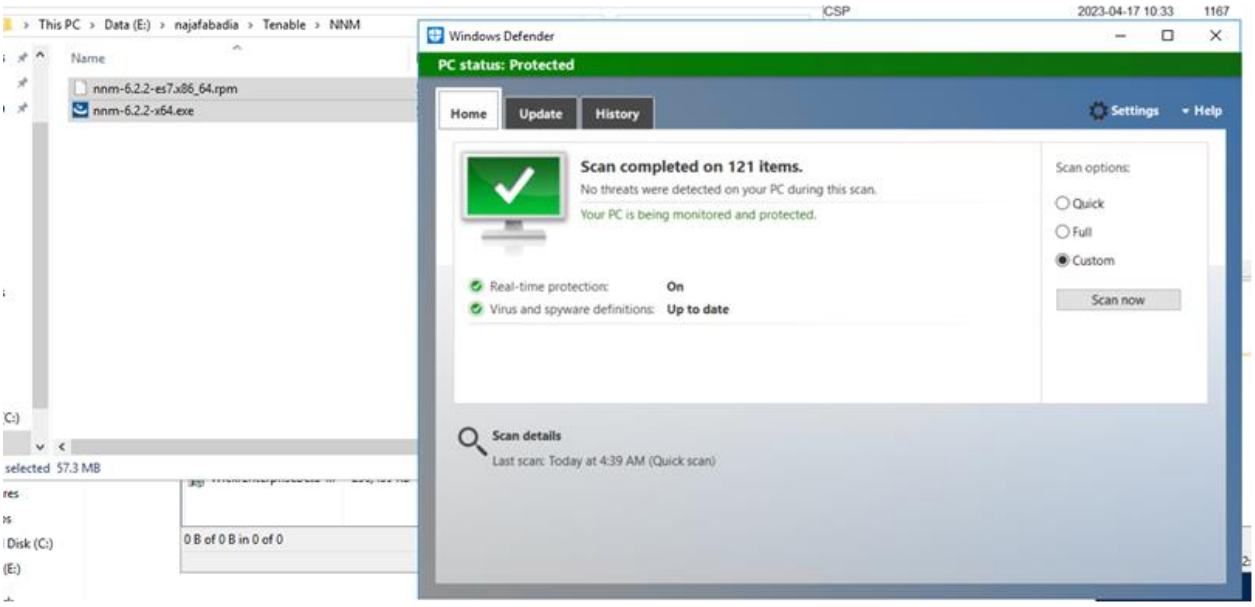
Listed below are the third-party libraries used by the TOE. Note that these libraries do not necessarily relate to the TOE functionality claimed in the Security Target; however, since they are bundled with the product itself they are disclosed since a vulnerability in outside the logical boundary of the product could still present an exploitable vulnerability.

Library	Version
c-ares	1.19.1
curl	8.1.2
dpdk	20.08.0
expat	2.5.0
Hyperscan	5.4.0
libbacktrace	1.0
libbzip2	1.0.8
libpcap	1.9.1
libpcre	8.44
libxml2	2.11.1
libxslt	1.1.37
libxmlsec	1.2.37
zlib	1.2.13
openssl	3.0.9
sqlite	3.40.1
winpcap	4.1.3
chosen	1.0.0

Library	Version
d3	3.4.8
DataTables	1.10.18
Handlebars	4.7.7
jQuery	3.6.4
jQuery Cookie	1.4.1
jQuery FileUpload	5.19.8
jQuery HotKeys	0.8
jQuery iFrame Transport Plugin	1.7
jQuery scroll.To	1.4.3
jQuery-Storage-API	1.7.3
jQuery TableSorter	0.11
jQuery tipsy	1.0.0a
jQuery UI	1.13.2
jQuery UI Touch Punch	0.2.3
mobile-detect.js	1.3.0
moment	2.29.4
spin.js	1.2.5
sugar	1.3
xml2json	0.9

No vulnerabilities were identified.

The evaluator also run a virus scanner with the most current virus definitions against the application files and verify that no files were flagged as malicious.



The single virus scan covered both the RHEL and Windows versions of the TOE.

3.5 Life-Cycle Support (ALC)

3.5.1 Labeling of the TOE (ALC_CMC.1)

3.5.1.1 Assurance Activity

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.

Section 1.1 of [ST] (“Security Target, TOE and CC Identification”) includes the TOE identification. The TOE is identified in terms of the software included in the evaluated configuration. The TOE is Nessus Network Monitor 6.2.2, supported on RHEL 8.7 and Windows Server 2019.

The TOE version is consistent with the version number of the TOE identified in [User] and the version identified by the TOE sample received for testing.

3.5.2 TOE Coverage (ALC_CMS.1)

3.5.2.1 Assurance Activity

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.

As described in Section 3.5.1 above, the evaluator confirmed the TOE is labelled with unique software version identifiers. Section 6.7 of [ST] ("Protection of the TSF") describes how each TOE version uses security features and APIs provided by its platform. This includes data execution protection, stack-based buffer overflow protection, and compatibility with platform security features.

3.5.3 Timely Security Update (ALC_TSU_EXT.1)

3.5.3.1 Assurance Activity

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.

Tenable supports a timely security update process for the TOE In addition to their own internal research, the product vendor supports disclosure of potential issues using community forums, direct engagement,

encrypted email (PGP) from customers and researchers, and the Tenable support channel. For issues where there is a potential security concern, the support channel uses HTTPS for secure disclosure.

When an issue is reported, Tenable will determine its applicability to the product. The length of time needed to make this determination depends on the complexity of the issue and the extent to which it can be reproduced; well-documented issues such as exposure to a published CVE can be made quickly. If found to be a security issue, a patch is released within 30 days. Tenable monitors the third-party components used by the TOE for potential security issues as well. However, an issue with a dependent component may not be addressed if found not to be applicable to the TOE. For example, security issues are frequently found within the PHP image library but Tenable does not install this library as part of the Nessus distribution.

Security updates to the TOE are delivered as regular update packages in the same manner as a functional update.