



www.GossamerSec.com

ASSURANCE ACTIVITY REPORT FOR PALO ALTO NETWORKS CORTEX XSOAR ENGINE 6.6

Version 0.3
09/30/2022

Prepared by:

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

Prepared for:

National Information Assurance Partnership
Common Criteria Evaluation and Validation Scheme



REVISION HISTORY

Revision	Date	Authors	Summary
Version 0.1	09/13/2022	Gossamer	Initial draft
Version 0.2	09/27/2022	Gossamer	Address NIAP comments
Version 0.3	09/30/2022	Gossamer	Address NIAP comments

The TOE Evaluation was Sponsored by:
Palo Alto Networks

Evaluation Personnel:

- Tammy Compton
- John Messiha

Common Criteria Versions:

- Common Criteria for Information Technology Security Evaluation Part 1: Introduction, Version 3.1, Revision 5, April 2017
- Common Criteria for Information Technology Security Evaluation Part 2: Security functional components, Version 3.1, Revision 5, April 2017
- Common Criteria for Information Technology Security Evaluation Part 3: Security assurance components, Version 3.1, Revision 5, April 2017

Common Evaluation Methodology Versions:

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



TABLE OF CONTENTS

- 1. Introduction5
 - 1.1 References.....5
 - 1.2 TOE CAVP Certificates5
 - 1.3 Test Equivalency.....6
- 2. Protection Profile SFR Assurance Activities7
 - 2.1 Cryptographic support (FCS)7
 - 2.1.1 Cryptographic Key Generation Services (ASPP14:FCS_CKM.1).....7
 - 2.1.2 Cryptographic Asymmetric Key Generation (ASPP14:FCS_CKM.1/AK).....7
 - 2.1.3 Cryptographic Key Establishment (ASPP14:FCS_CKM.2)11
 - 2.1.4 Cryptographic Operation - Hashing (ASPP14:FCS_COP.1/Hash).....14
 - 2.1.5 Cryptographic Operation - Keyed-Hash Message Authentication - per TD0626 (ASPP14:FCS_COP.1/KeyedHash)16
 - 2.1.6 Cryptographic Operation - Signing (ASPP14:FCS_COP.1/Sig)16
 - 2.1.7 Cryptographic Operation - Encryption/Decryption (ASPP14:FCS_COP.1/SKC).....17
 - 2.1.8 Random Bit Generation Services (ASPP14:FCS_RBG_EXT.1)23
 - 2.1.9 Random Bit Generation from Application (ASPP14:FCS_RBG_EXT.2)25
 - 2.1.10 Storage of Credentials (ASPP14:FCS_STO_EXT.1).....27
 - 2.1.11 TLS Protocol (PKGTLS11:FCS_TLS_EXT.1)28
 - 2.1.12 TLS Client Protocol (PKGTLS11:FCS_TLSC_EXT.1)29
 - 2.1.13 TLS Client Support for Mutual Authentication (PKGTLS11:FCS_TLSC_EXT.2).....35
 - 2.1.14 TLS Client Support for Supported Groups Extension (PKGTLS11:FCS_TLSC_EXT.5)37
 - 2.2 User data protection (FDP)37
 - 2.2.1 Encryption Of Sensitive Application Data (ASPP14:FDP_DAR_EXT.1)37
 - 2.2.2 Access to Platform Resources (ASPP14:FDP_DEC_EXT.1).....39
 - 2.2.3 Network Communications (ASPP14:FDP_NET_EXT.1)41
 - 2.3 Identification and authentication (FIA)42
 - 2.3.1 X.509 Certificate Validation (ASPP14:FIA_X509_EXT.1).....42
 - 2.3.2 X.509 Certificate Authentication (ASPP14:FIA_X509_EXT.2)46
 - 2.4 Security management (FMT).....48
 - 2.4.1 Secure by Default Configuration (ASPP14:FMT_CFG_EXT.1).....48



- 2.4.2 Supported Configuration Mechanism - per TD0624 (ASPP14:FMT_MEC_EXT.1).....50
- 2.4.3 Specification of Management Functions (ASPP14:FMT_SMF.1).....52
- 2.5 Privacy (FPR).....52
 - 2.5.1 User Consent for Transmission of Personally Identifiable (ASPP14:FPR_ANO_EXT.1)52
- 2.6 Protection of the TSF (FPT)53
 - 2.6.1 Anti-Exploitation Capabilities (ASPP14:FPT_AEX_EXT.1)53
 - 2.6.2 Use of Supported Services and APIs (ASPP14:FPT_API_EXT.1).....59
 - 2.6.3 Software Identification and Versions (ASPP14:FPT_IDV_EXT.1).....59
 - 2.6.4 Use of Third Party Libraries (ASPP14:FPT_LIB_EXT.1).....60
 - 2.6.5 Integrity for Installation and Update (ASPP14:FPT_TUD_EXT.1)60
 - 2.6.6 Integrity for Installation and Update - per TD0628 (ASPP14:FPT_TUD_EXT.2)63
- 2.7 Trusted path/channels (FTP).....65
 - 2.7.1 Protection of Data in Transit (ASPP14:FTP_DIT_EXT.1)65
- 3. Protection Profile SAR Assurance Activities68
 - 3.1 Development (ADV)68
 - 3.1.1 Basic Functional Specification (ADV_FSP.1).....68
 - 3.2 Guidance documents (AGD).....68
 - 3.2.1 Operational User Guidance (AGD_OPE.1)68
 - 3.2.2 Preparative Procedures (AGD_PRE.1).....69
 - 3.3 Life-cycle support (ALC).....69
 - 3.3.1 Labelling of the TOE (ALC_CMC.1)69
 - 3.3.2 TOE CM Coverage (ALC_CMS.1).....69
 - 3.3.3 Timely Security Updates (ALC_TSU_EXT.1).....70
 - 3.4 Tests (ATE).....71
 - 3.4.1 Independent Testing - Conformance (ATE_IND.1).....71
 - 3.5 Vulnerability assessment (AVA)72
 - 3.5.1 Vulnerability Survey (AVA_VAN.1).....72



1. INTRODUCTION

This document presents evaluations results of the Palo Alto Networks Cortex XSOAR Engine 6.6 ASPP14/PKGTLS11 evaluation. This document contains a description of the assurance activities and associated results as performed by the evaluators.

1.1 REFERENCES

The following evidence was used to complete the Assurance Activities:

- Palo Alto Networks Cortex XSOAR Engine 6.6 Security Target, Version 1.1, 09/30/2022 [ST]
- Palo Alto Networks Common Criteria Evaluated Configuration Guide (CCECG) Cortex XSOAR Server and Engine 6.6, 09/16/2022 [Admin Guide]

The following are the relevant protection profiles and functional packages used for this evaluation:

- Protection Profile for Application Software, Version 1.4, 07 October 2021 (ASPP14)
- Functional Package for Transport Layer Security (TLS), Version 1.1, 01 March 2019 (PKGTLS11)

1.2 TOE CAVP CERTIFICATES

The following table shows CAVP certificates for algorithms:

Function(s)	Standards	Certificates
Asymmetric key generation (FCS_CKM.1 and FCS_CKM.1/AK)		
ECDSA (P-256, P-384 curves)	FIPS PUB 186-4, “Digital Signature Standard (DSS)”, Appendix B.4	#A2517
Cryptographic key establishment (FCS_CKM.2)		
Elliptic curve-based scheme	NIST Special Publication 800-56A Revision 3, “Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography”	#A2517
Symmetric encryption/decryption (FCS_COP.1/SKC)		
AES CBC, GCM (128, 256 bits)	FIPS PUB 197 CBC as defined in NIST SP 800-38A GCM as defined in NIST SP 800-38D	#A2517
Cryptographic hashing (FCS_COP.1/Hash)		
SHA-1, SHA-256, SHA-384, SHA-512	FIPS PUB 180-4	#A2517
Cryptographic signature services (FCS_COP.1/Sig)		



RSA with 2048-bit modulus or 3072-bit modulus	FIPS PUB 186-4	#A2517
ECDSA with NIST Curves P-256, P-384	FIPS PUB 186-4	#A2517
Keyed-hash message authentication (FCS_COP.1/KeyedHash)		
HMAC-SHA-256	FIPS Pub 198-1	#A2517
HMAC-SHA-384	FIPS Pub 180-4	
HMAC-SHA-512		
Deterministic random bit generation (FCS_RBG_EXT.2)		
CTR_DRBG (AES) (256 bits)	NIST SP 800-90A	#A2517

Table 1 - TOE CAVP Certificates

1.3 TEST EQUIVALENCY

The TOE is compiled to execute on a Linux platform that meets the hardware requirements contained in the Security Target. Though several platforms are listed in the Security Target, Gossamer executed testing on a Red Hat Enterprise Linux (RHEL) Release 8.4 because the application is the same on all platforms. The TOE only relies on the software functions of each platform and not the hardware. Similarly, algorithm testing was performed on the same hardware platform used for CCTL testing. Since the exact same application runs on all platforms, only one set of algorithm certificates was needed. Please refer to the AAR for the table of algorithm certificates.



2. PROTECTION PROFILE SFR ASSURANCE ACTIVITIES

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

2.1 CRYPTOGRAPHIC SUPPORT (FCS)

2.1.1 CRYPTOGRAPHIC KEY GENERATION SERVICES (ASPP14:FCS_CKM.1)

2.1.1.1 ASPP14:FCS_CKM.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

The application needs asymmetric key generation services according to its ST description. See ASPP14:FCS_CKM.1.1/AK for a discussion of the asymmetric key generation.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.1.2 CRYPTOGRAPHIC ASYMMETRIC KEY GENERATION (ASPP14:FCS_CKM.1/AK)

2.1.2.1 ASPP14:FCS_CKM.1.1/AK

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined



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

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

Section 6.1 of the ST states that during a TLS handshake, the TOE utilizes ECC scheme to generate asymmetric keys with NIST curves that include P-256 and P-384 that adhere to the NIST Special Publication 186-4.

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

The key generation scheme(s) and key size(s) are not configurable. During a TLS handshake, the TOE utilizes ECC scheme to generate asymmetric keys with NIST curves that include P-256 and P-384.

The "Verify FIPS-CC Mode" section of the [Admin Guide] indicates that the FIPS-CC mode cannot be disabled, it requires all power-up self-tests to pass. If any of the self-tests fails, the system will not start up (i.e., in FIPS error state). If this occurs, uninstalling and re-installing the application is required. The FIPS-CC mode refers to a set of features (one of which is to enforce the TOE to use the selected key generation scheme(s) and key size(s)) which meet the requirements for this evaluation and ensures that the TOE installation will operate in a manner consistent with the CC evaluation.

Component Testing Assurance Activities: If the application 'implements asymmetric key generation,' then the following test activities shall be carried out. Evaluation Activity Note: The following tests may require the developer to provide access to a developer environment that provides the evaluator with tools that are typically available to end-users of the application. Key Generation for FIPS PUB 186-4 RSA Schemes The evaluator shall verify the implementation of RSA Key Generation by the TOE using the Key Generation test. This test verifies the ability of the TSF to correctly produce values for the key components including the public verification exponent e , the private prime factors p and q , the public modulus n and the calculation of the private signature exponent d . Key Pair generation specifies 5 ways (or methods) to generate the primes p and q . These include:

1. Random Primes:

Provable primes

Probable primes

2. Primes with Conditions:

Primes p_1, p_2, q_1, q_2, p and q shall all be provable primes

Primes p_1, p_2, q_1 , and q_2 shall be provable primes and p and q shall be probable primes



Primes p_1, p_2, q_1, q_2, p and q shall all be probable primes

To test the key generation method for the Random Provable primes method and for all the Primes with Conditions methods, the evaluator must seed the TSF key generation routine with sufficient data to deterministically generate the RSA key pair. This includes the random seed(s), the public exponent of the RSA key, and the desired key length. For each key length supported, the evaluator shall have the TSF generate 25 key pairs. The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated from a known good implementation. If possible, the Random Probable primes method should also be verified against a known good implementation as described above. Otherwise, the evaluator shall have the TSF generate 10 keys pairs for each supported key length $nlen$ and verify:

$$n = p * q,$$

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

$$\text{GCD}(p-1, e) = 1,$$

$$\text{GCD}(q-1, e) = 1,$$

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

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

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

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

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

$$e * d = 1 \text{ mod } \text{LCM}(p-1, q-1).$$

Key Generation for Elliptic Curve Cryptography (ECC)

FIPS 186-4 ECC Key Generation Test For each supported NIST curve, i.e., P-256, P384 and P-521, the evaluator shall require the implementation under test (IUT) to generate 10 private/public key pairs. The private key shall be generated using an approved random bit generator (RBG). To determine correctness, the evaluator shall submit the generated key pairs to the public key verification (PKV) function of a known good implementation.

FIPS 186-4 Public Key Verification (PKV) Test For each supported NIST curve, i.e., P256, P-384 and P-521, the evaluator shall generate 10 private/public key pairs using the key generation function of a known good implementation and modify five of the public key values so that they are incorrect, leaving five values unchanged (i.e., correct). The evaluator shall obtain in response a set of 10 PASS/FAIL values.

Key Generation for Finite-Field Cryptography (FFC) The evaluator shall verify the implementation of the Parameters Generation and the Key Generation for FFC by the TOE using the Parameter Generation and Key Generation test. This test verifies the ability of the TSF to correctly produce values for the field prime p , the cryptographic prime q (dividing $p-1$), the cryptographic group generator g ,



and the calculation of the private key x and public key y . The Parameter generation specifies 2 ways (or methods) to generate the cryptographic prime q and the field prime p :

Cryptographic and Field Primes:

Primes q and p shall both be provable primes

Primes q and field prime p shall both be probable primes

and two ways to generate the cryptographic group generator g :

Cryptographic Group Generator:

Generator g constructed through a verifiable process

Generator g constructed through an unverifiable process.

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

$\text{len}(q)$ bit output of RBG where $1 = x = q-1$

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

The security strength of the RBG must be at least that of the security offered by the FFC parameter set. To test the cryptographic and field prime generation method for the provable primes method and/or the group generator g for a verifiable process, the evaluator must seed the TSF parameter generation routine with sufficient data to deterministically generate the parameter set. For each key length supported, the evaluator shall have the TSF generate 25 parameter sets and key pairs. The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated from a known good implementation. Verification must also confirm

$g \neq 0, 1$

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 has been CAVP tested. Refer to the CAVP certificates identified in the “TOE CAVP Certificates” table in Section 1.2.

2.1.3 CRYPTOGRAPHIC KEY ESTABLISHMENT (ASPP14:FCS_CKM.2)

2.1.3.1 ASPP14:FCS_CKM.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.1 of the ST states the TOE implements key establishment methods using elliptical curve key establishment scheme (ECDHE). The curves utilized by the TOE include P-256 and P-384 as defined in NIST SP 800-56A (revision 3). This corresponds to the key generation schemes identified in FCS_CKM.1.1.

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

The [Admin Guide] does not contain any configuration of the modes and key sizes because the TOE negotiates the claimed ciphers by default. As stated by the ST, the TOE only implements key establishment methods using elliptical curve key establishment scheme (ECDHE).

Component Testing Assurance Activities: Evaluation Activity Note: The following tests require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on factory products.

Key Establishment Schemes

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

SP800-56A Key Establishment Schemes

The evaluator shall verify a TOE's implementation of SP800-56A key agreement schemes using the following Function and Validity tests. These validation tests for each key agreement scheme verify that a TOE has implemented the components of the key agreement scheme according to the specifications in the Recommendation. These components include the calculation of the DLC primitives (the shared secret value Z) and the calculation of the derived keying material (DKM) via the Key Derivation Function (KDF). If key confirmation is



supported, the evaluator shall also verify that the components of key confirmation have been implemented correctly, using the test procedures described below. This includes the parsing of the DKM, the generation of MACdata and the calculation of MACtag.

Function Test

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

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

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

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

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

Validity Test

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

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



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

SP800-56B Key Establishment Schemes

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

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

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

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

To conduct this test the evaluator shall generate or obtain test vectors from a known good implementation of the TOE supported schemes. For each combination of supported key establishment scheme and its options (with 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 private key, the plaintext keying material (KeyData), any additional input parameters if applicable, the MacTag in cases where key confirmation is incorporated, and the outputted ciphertext. For each test vector, the evaluator shall perform the key establishment decryption operation on the TOE and ensure that the outputted plaintext keying material (KeyData) is equivalent to the plaintext keying material in the test vector. In cases where key confirmation is incorporated, the evaluator shall perform the key confirmation steps and ensure that the outputted MacTag is equivalent to the MacTag in the test vector.

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



checks described in NIST Special Publication 800-56B section 7.2.3.3, ensure that each decryption attempt results in an error, and ensure that any outputted or logged error message is identical for each.

RSA-based key establishment

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

Diffie-Hellman Group 14

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

FFC Schemes using 'safe-prime' groups

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

The TOE has been CAVP tested. Refer to the CAVP certificates identified in the table "TOE CAVP Certificates" in Section 1.2.

2.1.4 CRYPTOGRAPHIC OPERATION - HASHING (ASPP14:FCS_COP.1/HASH)

2.1.4.1 ASPP14:FCS_COP.1.1/HASH

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.1 of the ST states that the TOE uses hash functions that include SHA-1, SHA-256, SHA-384, and SHA-512 as defined in FIPS 180-4. The digest sizes include 160-bits, 256-bits, 384-bits, and 512-bits that are compliant with FIPS 180-4. The hashing capabilities are utilized for digital signature verification and generation and software integrity checks. SHA-1 is not used for generating digital signatures as noted in SP 800-131A but is used only for verification for legacy purpose. The TOE uses SHA-256 and SHA-384 hashing as part of generating digital signatures. SHA-512 is used as part of the software integrity power-up test.

Component Guidance Assurance Activities: None Defined



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

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

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

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

Test 3: Selected Long Messages Test - Bit oriented Mode The evaluators devise an input set consisting of m messages, where m is the block length of the hash algorithm. The length of the i th message is $512 + 99*i$, where $1 \leq i \leq m$. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

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 i th message is $512 + 8*99*i$, where $1 \leq i \leq m/8$. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

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 has been CAVP tested. Refer to the CAVP certificates identified in the table "TOE CAVP Certificates" in Section 1.2.



2.1.5 CRYPTOGRAPHIC OPERATION - KEYED-HASH MESSAGE AUTHENTICATION - PER TD0626 (ASPP14:FCS_COP.1/KEYEDHASH)

2.1.5.1 ASPP14:FCS_COP.1.1/KEYEDHASH

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

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

The TOE has been CAVP tested. Refer to the CAVP certificates identified in the table “TOE CAVP Certificates” in Section 1.2.

2.1.6 CRYPTOGRAPHIC OPERATION - SIGNING (ASPP14:FCS_COP.1/Sig)

2.1.6.1 ASPP14:FCS_COP.1.1/Sig

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

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

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

ECDSA Algorithm Tests



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

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

RSA Signature Algorithm Tests

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

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

The TOE has been CAVP tested. Refer to the CAVP certificates identified in the table "TOE CAVP Certificates" in Section 1.2.

2.1.7 CRYPTOGRAPHIC OPERATION - ENCRYPTION/DECRYPTION (ASPP14:FCS_COP.1/SKC)

2.1.7.1 ASPP14:FCS_COP.1.1/SKC

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

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



The [Admin Guide] does not contain any configuration of the modes and key sizes because the TOE negotiates the claimed ciphers by default.

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

AES-CBC Known Answer Tests

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

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

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

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

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

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



AES-CBC Multi-Block Message Test

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

AES-CBC Monte Carlo Tests

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

Input: PT, IV, Key

for $i = 1$ to 1000:

if $i == 1$:

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

PT = IV

else:

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

PT = CT[i-1]

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

AES-GCM Monte Carlo Tests

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

128 bit and 256 bit keys



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

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

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

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

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

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

AES-XTS Tests

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

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

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

Using a set of 100 (key, plaintext and 128-bit random tweak value) 3-tuples and obtain the ciphertext that results from XTS-AES encrypt. The evaluator may supply a data unit sequence number instead of the tweak value if the implementation supports it. The data unit sequence number is a base-10 number ranging between 0 and 255 that implementations convert to a tweak value internally.

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

AES-CCM Tests

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



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

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

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

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

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

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

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

Variable Associated Data Test

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

Variable Payload Test

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

Variable Nonce Test

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

Variable Tag Test

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

Decryption-Verification Process Test

To test the decryption-verification functionality of AES-CCM, for each combination of supported associated data length, payload length, nonce length, and tag length, the evaluator shall supply a key value and 15 sets of input



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

AES-CTR Tests

Test 1: Known Answer Tests (KATs)

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

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

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

To test the encrypt functionality, the evaluator shall supply the two sets of key values described below and obtain the ciphertext values that result from AES encryption of an all zeros plaintext using the given key values and 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 $1 \leq i \leq 10$. 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 $1 \leq i \leq 10$. 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:

$CT[i] = \text{AES-ECB-Encrypt}(\text{Key}, \text{PT})$

$\text{PT} = CT[i]$

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 has been CAVP tested. Refer to the CAVP certificates identified in the table “TOE CAVP Certificates” in Section 1.2.

2.1.8 RANDOM BIT GENERATION SERVICES (ASPP14:FCS_RBG_EXT.1)

2.1.8.1 ASPP14:FCS_RBG_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined



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

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

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

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

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

Section 5.2 of the ST selects 'implement DRBG functionality'. The evaluator verified that FCS_RBG_EXT.2 is also included in the ST.

Component Guidance Assurance Activities: None Defined

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

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

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

Platforms: Android....

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

Platforms: Microsoft Windows....

The evaluator shall verify that rand_s, RtlGenRandom, BCryptGenRandom, or CryptGenRandom API is used for classic desktop applications. The evaluator shall verify the application uses the RNGCryptoServiceProvider class or derives a class from System.Security.Cryptography.RandomNumberGenerator API for Windows Universal



Applications. It is only required that the API is called/invoked, there is no requirement that the API be used directly. In future versions of this document, CryptGenRandom may be removed as an option as it is no longer the preferred API per vendor documentation.

Platforms: Apple iOS....

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

Platforms: Linux....

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

Platforms: Oracle Solaris....

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

Platforms: Apple macOS....

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

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

This assurance activity does not apply as the TOE does not invoke platform provided random bit generation services.

2.1.9 RANDOM BIT GENERATION FROM APPLICATION (ASPP14:FCS_RBG_EXT.2)

2.1.9.1 ASPP14:FCS_RBG_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

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



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

Test 2: The evaluators shall perform a Monte Carlo Test. For this test, they supply an initial Seed and DT value to the TSF RBG function; each of these is 128 bits. The evaluators shall also provide a key (of the length appropriate to the AES algorithm) that is constant throughout the test. The evaluators then invoke the TSF RBG 10,000 times, with the DT value being incremented by 1 on each iteration, and the new seed for the subsequent iteration produced as specified in 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 to 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 to 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 than or



equal to seed length. If the implementation only supports one personalization string length, then the same length can be used for both values. If more than one string length is support, the evaluator 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 has been CAVP tested. Refer to the CAVP certificates identified in the table “TOE CAVP Certificates” in Section 1.2.

2.1.9.2 ASPP14:FCS_RBG_EXT.2.2

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

Palo Alto has delivered an Entropy Analysis to NIAP that contained the required documentation.

Guidance Assurance Activities: None Defined

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

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.1.10 STORAGE OF CREDENTIALS (ASPP14:FCS_STO_EXT.1)

2.1.10.1 ASPP14:FCS_STO_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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



Not Applicable as the selection in the ST is “not store any credentials”.

Component Guidance Assurance Activities: None Defined

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

Platforms: Android....

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

Platforms: Microsoft Windows....

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

Platforms: Apple iOS....

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

Platforms: Linux....

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

Platforms: Oracle Solaris....

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

Platforms: Apple macOS....

The evaluator shall verify that all credentials are stored within Keychain

Not Applicable as the TOE does not store any credentials.

2.1.11 TLS PROTOCOL (PKG TLS 1.1:FCS_TLS_EXT.1)

2.1.11.1 PKG TLS 1.1:FCS_TLS_EXT.1.1

TSS Assurance Activities: None Defined



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

Section 5.2 of the ST selects TLS Client. The ST contains the requirement accordingly.

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.1.12 TLS CLIENT PROTOCOL (PKG TLS11:FCS_TLSC_EXT.1)

2.1.12.1 PKG TLS11:FCS_TLSC_EXT.1.1

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

Section 6.1 of the ST provides a list of cipher suites supported by the TOE in its TLS client Implementation. The list is as follows and matches the requirement:

- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289

The ciphersuites cannot be configured by the administrator and the TOE enforces this by default.

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

The "Storing CA certificates in Linux" section in the [Admin Guide] describes how to upload a CA certificate to the Cortex XSOAR trust anchor. The "TLS/X509 Configuration" section in the [Admin Guide] provides instructions for configuring TLS and trust certificates for connections to external services.

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

Test 1: The evaluator shall establish a TLS connection using each of the cipher suites specified by the requirement. This connection may be established as part of the establishment of a higher-level protocol, e.g., as part of an EAP session. It is sufficient to observe the successful negotiation of a cipher suite to satisfy the intent of the test; it is



not necessary to examine the characteristics of the encrypted traffic in an attempt to discern the cipher suite being used (for example, that the cryptographic algorithm is 128-bit AES and not 256-bit AES).

Test 2: The goal of the following test is to verify that the TOE accepts only certificates with appropriate values in the extendedKeyUsage extension, and implicitly that the TOE correctly parses the extendedKeyUsage extension as part of X.509v3 server certificate validation. The evaluator shall attempt to establish the connection using a server with a server certificate that contains the Server Authentication purpose in the extendedKeyUsage extension and verify that a connection is established. The evaluator shall repeat this test using a different, but otherwise valid and trusted, certificate that lacks the Server Authentication purpose in the extendedKeyUsage extension and ensure that a connection is not established. Ideally, the two certificates should be similar in structure, the types of identifiers used, and the chain of trust.

Test 3: The evaluator shall send a server certificate in the TLS connection that does not match the server-selected cipher suite (for example, send a ECDSA certificate while using the TLS_RSA_WITH_AES_128_CBC_SHA cipher suite or send a RSA certificate while using one of the ECDSA cipher suites.) The evaluator shall verify that the product disconnects after receiving the server's Certificate handshake message.

Test 4: The evaluator shall configure the server to select the TLS_NULL_WITH_NULL_NULL cipher suite and verify that the client denies the connection.

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

Test 5.1: Change the TLS version selected by the server in the Server Hello to an undefined TLS version (for example 1.5 represented by the two bytes 03 06) and verify that the client rejects the connection.

Test 5.2: Change the TLS version selected by the server in the Server Hello to the most recent unsupported TLS version (for example 1.1 represented by the two bytes 03 02) and verify that the client rejects the connection.

Test 5.3: [conditional] If DHE or ECDHE cipher suites are supported, modify at least one byte in the server's nonce in the Server Hello handshake message, and verify that the client does not complete the handshake and no application data flows.

Test 5.4: Modify the server's selected cipher suite in the Server Hello handshake message to be a cipher suite not presented in the Client Hello handshake message. The evaluator shall verify that the client does not complete the handshake and no application data flows.

Test 5.5: [conditional] If DHE or ECDHE cipher suites are supported, modify the signature block in the server's Key Exchange handshake message, and verify that the client does not complete the handshake and no application data flows. This test does not apply to cipher suites using RSA key exchange. If a TOE only supports RSA key exchange in conjunction with TLS, then this test shall be omitted.

Test 5.6: Modify a byte in the Server Finished handshake message, and verify that the client does not complete the handshake and no application data flows.



Test 5.7: Send a message consisting of random bytes from the server after the server has issued the Change Cipher Spec message and verify that the client does not complete the handshake and no application data flows. The message must still have a valid 5-byte record header in order to ensure the message will be parsed as TLS.

The evaluator performed the following tests against the Cortex XSOAR Engine's TLS client interface.

Test 1: The evaluator established a TLS session using the interface stated above for each of the claimed ciphersuites in turn. The evaluator used a network sniffer to capture the TLS session negotiation and observed that each expected TLS cipher is negotiated.

Test 2: The evaluator established a TLS session using the interface stated above. The evaluator configured the server to send a certificate with the Server Authentication purpose in the extendedKeyUsage field. Using a network sniffer, the evaluator captured the TLS session negotiation and observed that the TLS session was successfully negotiated. The evaluator reconfigured the test server to retry the TLS session using a certificate that is missing the Server Authentication purpose in the extendedKeyUsage field. Using a network sniffer, the evaluator captured the TLS session negotiation and observed that the TLS session is not successfully negotiated.

Test 3: The evaluator established a TLS session using the interface stated above. A modified test server negotiates an ECDSA ciphersuite, but returns an RSA certificate. Using a network sniffer to capture the TLS session negotiation and observed that the TLS session is not negotiated successfully.

Test 4: The evaluator configured a test server to accept only the TLS_NULL_WITH_NULL_NULL ciphersuite. The evaluator then attempted to establish a TLS session using the interface stated above to that test server. Using a network sniffer, the evaluator captured the TLS session negotiation and observed that the TLS session is not successfully negotiated.

Test 5: The evaluator obtained a packet captures of the TLS session negotiation using the interface stated above and a test server with Mutual Authentication configured on the test server. The evaluator made connection attempts from the client to the test server. The server implementation of the TLS protocol was modified as stated in the 7 scenarios described by the Assurance Activity. The evaluator inspected each packet captures to ensure that the connections are rejected for each scenario.

2.1.12.2 PKGTLS11:FCS_TLSC_EXT.1.2

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

Section 6.1 explains that all data that is transmitted between the TOE and its' Cortex XSOAR Server is encrypted using TLSv1.2. When the TOE is establishing a TLS session, it checks the reference identifier that has been specified by the administrators. These reference identifiers can include FQDN or IP address (not recommended) and are



checked by looking at the Subject Alternative Name (SAN). The certificate must have a SAN field, or the TLS connection will be terminated. The TOE does not support wildcards if a certificate is presented with one in it. Certificate pinning is also not supported.

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

The section entitled "Storing CA certificates in Linux" in the [Admin Guide] provides instructions to configure a Local CA certificate into the Cortex XSOAR trust anchor. Section "TLS/X509 Configuration" in the [Admin Guide] contains instructions for an administrator to configure the SAN trust list for the agent certificates. Additionally, there is a table in that same section in the [Admin Guide] containing the complete set of security relevant variables that could be configured.

Testing Assurance Activities: The evaluator shall configure the reference identifier according to the AGD guidance and perform the following tests during a TLS connection. If the TOE supports certificate pinning, all pinned certificates must be removed before performing Tests 1 through 6. A pinned certificate must be added prior to performing Test 7. (TD0499 applied)

Test 1: The evaluator shall present a server certificate that contains a CN that does not match the reference identifier and does not contain the SAN extension. The evaluator shall verify that the connection fails. Note that some systems might require the presence of the SAN extension. In this case the connection would still fail but for the reason of the missing SAN extension instead of the mismatch of CN and reference identifier. Both reasons are acceptable to pass Test 1.

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

Test 3: [conditional] If the TOE does not mandate the presence of the SAN extension, the evaluator shall present a server certificate that contains a CN that matches the reference identifier and does not contain the SAN extension. The evaluator shall verify that the connection succeeds. If the TOE does mandate the presence of the SAN extension, this Test shall be omitted.

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

Test 5: The evaluator shall perform the following wildcard tests with each supported type of reference identifier. The support for wildcards is intended to be optional. If wildcards are supported, the first, second, and third tests below shall be executed. If wildcards are not supported, then the fourth test below shall be executed.



Test 5.1: [conditional]: If wildcards are supported, the evaluator shall present a server certificate containing a wildcard that is not in the left-most label of the presented identifier (e.g. foo.*.example.com) and verify that the connection fails.

Test 5.2: [conditional]: If wildcards are supported, the evaluator shall present a server certificate containing a wildcard in the left-most label but not preceding the public suffix (e.g. *.example.com). The evaluator shall configure the reference identifier with a single left-most label (e.g. foo.example.com) and verify that the connection succeeds. The evaluator shall configure the reference identifier without a left-most label as in the certificate (e.g. example.com) and verify that the connection fails. The evaluator shall configure the reference identifier with two left-most labels (e.g. bar.foo.example.com) and verify that the connection fails.

Test 5.3: [conditional]: If wildcards are supported, the evaluator shall present a server certificate containing a wildcard in the left-most label immediately preceding the public suffix (e.g. *.com). The evaluator shall configure the reference identifier with a single left-most label (e.g. foo.com) and verify that the connection fails. The evaluator shall configure the reference identifier with two left-most labels (e.g. bar.foo.com) and verify that the connection fails.

Test 5.4: [conditional]: If wildcards are not supported, the evaluator shall present a server certificate containing a wildcard in the left-most label (e.g. *.example.com). The evaluator shall configure the reference identifier with a single left-most label (e.g. foo.example.com) and verify that the connection fails.

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

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

The evaluator performed the following tests against the Cortex XSOAR Engine's TLS client interface.

Test 1: The evaluator established a TLS session using the interface stated above targeting a server using a valid certificate with a CN matching the domain name used by the client. Using a network sniffer to capture the TLS session negotiation the evaluator examined the traffic capture and observed a successful connection. The evaluator then established a TLS session targeting a server using a server certificate that does not contain an identifier in either the Subject Alternative Name (SAN) or Common Name (CN) that matches the reference identifier. Using a network sniffer to capture the TLS session negotiation the evaluator examined the traffic capture and observed that the TLS session was not negotiated successfully.

Test 2: The evaluator established a TLS session using the interfaces stated above targeting a server using a server certificate that contains a CN that matches the reference identifier, contains the SAN extension, but does not contain an identifier in the SAN that matches the reference identifier. Using a network sniffer to capture the TLS



session negotiation the evaluator examined the traffic capture and observed that the TLS session was not negotiated successfully.

Test 3: The evaluator established a TLS session using the interfaces stated above targeting a server using a server certificate that contains a CN that matches the reference identifier and does not contain the SAN extension. Using a network sniffer to capture the TLS session negotiation the evaluator examined the traffic capture and observed that the TLS session was negotiated successfully.

Test 4: The evaluator established a TLS session using the interfaces stated above targeting a server using a server certificate that contains a CN that does not match the reference identifier but does contain an identifier in the SAN that matches. Using a network sniffer to capture the TLS session negotiation the evaluator examined the traffic capture and observed that the TLS session was negotiated successfully.

Test 5: Since the TOE does not support wildcards, the evaluator established a TLS session using the interfaces stated above targeting a server using a server certificate containing a wildcard in the left-most label (e.g. *.example.com). The evaluator configured the reference identifier with a single left-most label (e.g. foo.example.com) and verified that the connection failed.

Test 6: The TOE does not support the optional URI or Service Name used as reference identifiers.

Test 7: The TOE does not support certificate pinning.

2.1.12.3 PKGTLS11:FCS_TLSC_EXT.1.3

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

Not applicable as the override selection was not made.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator shall demonstrate that using an invalid certificate results in the function failing as follows, unless excepted:

Test 1a: The evaluator shall demonstrate that a server using a certificate with a valid certification path successfully connects.

Test 1b: The evaluator shall modify the certificate chain used by the server in test 1a to be invalid and demonstrate that a server using a certificate without a valid certification path to a trust store element of the TOE results in an authentication failure.



Test 1c [conditional]: If the TOE trust store can be managed, the evaluator shall modify the trust store element used in Test 1a to be untrusted and demonstrate that a connection attempt from the same server used in Test 1a results in an authentication failure.

(TD0513 applied)

Test 2: The evaluator shall demonstrate that a server using a certificate which has been revoked results in an authentication failure.

Test 3: The evaluator shall demonstrate that a server using a certificate which has passed its expiration date results in an authentication failure.

Test 4: The evaluator shall demonstrate that a server using a certificate which does not have a valid identifier results in an authentication failure.

The evaluator performed the following tests against the Cortex XSOAR Engine's TLS client interface.

Test 1: The evaluator configured a test server using a certificate with a valid certification path terminating in a certificate which was not configured in the TOE as trusted. The evaluator observed that the TOE rejected the certificate. The evaluator then loaded the trusted CA certificate(s) needed to validate the server's certificate, and demonstrated that the connection succeeded. The evaluator then deleted the CA certificate that was loaded in the previous test part, and showed that the connection again failed.

Test 2: The evaluator demonstrated that a server presenting a certificate which has been revoked to the TOE results in the TOE rejecting the certificate and the connection failing.

Test 3: The evaluator demonstrated that a server presenting a certificate which has passed its expiration date results in the TOE rejecting the certificate and the connection failing.

Test 4: The evaluator demonstrated that a server presenting a certificate which does not have a valid identifier results in the TOE rejecting the certificate and the connection failing.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.1.13 TLS CLIENT SUPPORT FOR MUTUAL AUTHENTICATION (PKGTLS11:FCS_TLSC_EXT.2)

2.1.13.1 PKGTLS11:FCS_TLSC_EXT.2.1



TSS Assurance Activities: 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. The evaluator shall also ensure that the TSS describes any factors beyond configuration that are necessary in order for the client to engage in mutual authentication using X.509v3 certificates.

Section 6.1 of the ST states that during the negotiation of the TLS handshake, X.509v3 certificates are used to verify the server's identity, and if configured, the client's identity. Additionally, client certificates should be configured on the TOE platform to support mutual authentication.

Guidance Assurance Activities: The evaluator shall ensure that the AGD guidance includes any instructions necessary to configure the TOE to perform mutual authentication. The evaluator also 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 "Configure Mutual Authentication for the Web UI" section in the [Admin Guide] includes instructions to configure the TOE to perform mutual authentication. The "TLS/X509 Configuration" section in the [Admin Guide] includes instructions for configuring the client-side certificates for TLS mutual authentications as well as examples for client-side certificates.

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

Test 1: The evaluator shall establish a connection to a server that is not configured for mutual authentication (i.e. does not send Server's Certificate Request (type 13) message). The evaluator observes negotiation of a TLS channel and confirms that the TOE did not send Client's Certificate message (type 11) during handshake.

Test 2: The evaluator shall establish a connection to a server with a shared trusted root that is configured for mutual authentication (i.e. it sends Server's Certificate Request (type 13) message). The evaluator observes negotiation of a TLS channel and confirms that the TOE responds with a non-empty Client's Certificate message (type 11) and Certificate Verify (type 15) message.

The evaluator performed the following tests against the Cortex XSOAR Engine's TLS client interface.

Test 1: The evaluator established a TLS session using the interface stated above and a test server that was not configured for mutual authentication. The evaluator observed that the TLS connection was successful and the TOE did not send a certificate or a certificate verify message.

Test 2: The evaluator established a TLS session using the interface stated above and a test server that was configured for mutual authentication. The evaluator observed that the TLS connection was successful and the TOE did send both a certificate and a certificate verify message.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined



2.1.14 TLS CLIENT SUPPORT FOR SUPPORTED GROUPS EXTENSION (PKG TLS11:FCS_TLSC_EXT.5)

2.1.14.1 PKG TLS11:FCS_TLSC_EXT.5.1

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

Section 6.1 of the ST states for the cipher suites noted below, the client hello extension supports secp256r1, secp384r1, and secp521r1 NIST curves.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator shall also perform the following test:

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

The evaluator performed the following tests against the Cortex XSOAR Engine's TLS client interface.

The evaluator configured a server to use various certificates necessary to cause a TLS negotiation using each of the key exchange methods. The evaluator then initiated a TLS session using the interfaces stated above while capturing traffic. Inspection of the traffic indicated that the TOE did in fact negotiate a successful connection using each of the claimed key exchange methods.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2 USER DATA PROTECTION (FDP)

2.2.1 ENCRYPTION OF SENSITIVE APPLICATION DATA (ASPP14:FDP_DAR_EXT.1)

2.2.1.1 ASPP14:FDP_DAR_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined



Testing Assurance Activities: None Defined

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

Section 6.2 of the ST states that the TOE does not store any sensitive data.

Component Guidance Assurance Activities: None Defined

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

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

Platforms: Android....

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

Platforms: Microsoft Windows....

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

Platforms: Apple iOS....

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

Platforms: Linux....

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

Platforms: Oracle Solaris....



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

Platforms: Apple macOS....

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

Not applicable as the TOE does not store any sensitive data.

2.2.2 ACCESS TO PLATFORM RESOURCES (ASPP14:FDP_DEC_EXT.1)

2.2.2.1 ASPP14:FDP_DEC_EXT.1.1

TSS Assurance Activities: None Defined

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

The TOE accesses the network connectivity of its platform as indicated by the SFR. This enables the TOE to provide the ability to communicate to its' Server.

Testing Assurance Activities: Platforms: Android....

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

Platforms: Microsoft Windows....

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

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

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

Platforms: Apple iOS....



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

Platforms: Linux....

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

Platforms: Oracle Solaris....

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

Platforms: Apple macOS....

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

The TOE accesses the network connectivity of its platform as indicated by the SFR. This enables the TOE to provide the ability to communicate to its' Server.

2.2.2.2 ASPP14:FDP_DEC_EXT.1.2

TSS Assurance Activities: None Defined

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

Not Applicable as the TOE does not access any sensitive information repositories.

Testing Assurance Activities: Platforms: Android....

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

Platforms: Microsoft Windows....

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

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



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

Platforms: Apple iOS....

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

Platforms: Linux....

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

Platforms: Oracle Solaris....

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

Platforms: Apple macOS....

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

Not Applicable as the TOE does not access any sensitive information repositories.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.3 NETWORK COMMUNICATIONS (ASPP14:FDP_NET_EXT.1)

2.2.3.1 ASPP14:FDP_NET_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined



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

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

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

Platforms: Android....

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

Test 1: The TOE uses TLS to communicate with its' Cortex XSOAR Server. The DTR evidence under FDP_NET_EXT.1.1 contains a packet capture for the only channel that the TOE uses to communicate, port 443 TLS. The evaluator verified that the communication protocols between the TOE and its' Cortex XSOAR Server are protected with TLS.

The evaluator set up a successful connection between the TOE (Engine) and its' Server. The packet capture showed that the communication between the TOE (Engine) and its' Server is protected with TLS.

Test 2: The evaluator used nmap to scan all TCP ports of the TOE while it was running. The results showed that nmap detected no open ports except for the specified TCP port 443 TLS that is used for communication. The evaluator then used nmap again to scan all UDP ports of the TOE while it was running. The results showed that nmap detected no open ports.

2.3 IDENTIFICATION AND AUTHENTICATION (FIA)

2.3.1 X.509 CERTIFICATE VALIDATION (ASPP14:FIA_X509_EXT.1)

2.3.1.1 ASPP14:FIA_X509_EXT.1.1

TSS Assurance Activities: 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.

Section 6.3 of the ST states the TOE implements the ability to perform certificate path validation on the certificate chain that is presented to it by the peer. The certificate path validation begins with the identity certificate presented



by the peer, and then proceeds in checking the intermediate CA certificate(s) until it reaches the trusted CA certificate stored in the platform OS trust store. If no trusted CA certificate is found (i.e., the TOE cannot choose to use any certificate for path validation), the session will terminate. The following steps are performed for each certificate in the path:

- The public key algorithm/parameters are checked (i.e., RSA/ECDSA key sizes meet requirements of 2048 bits or greater for RSA and 256 bits or greater for ECDSA)
- The certificate is checked to make sure it is not expired (i.e., validity period of the certificate must be proper)
- The certificate is checked to make sure it is not revoked using OCSP
- The issuer name is checked to ensure that it matches the subject name of the previous certificate in the chain
- The certificate is checked that it terminates with a trusted CA certificate and that all CA certificate have the basicConstraints extension present (and set to TRUE)
- The extendedKeyUsage field is checked such that OCSP certificates and server/client certificates contain the correct OID (e.g., OCSP Signing purpose and Server/Client Authentication purpose)
- The key usage extension of the certificate is checked to make sure that it is allowed to sign certificates

Guidance Assurance Activities: None Defined

Testing Assurance Activities: 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.

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



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 RFC6066 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.

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

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.)

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.)

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.)

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.

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 8 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 8, 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.

The TOE validates X509 public key certificates as a TLS client when communicating with the Cortex XSOAR Server. The evaluator performed the following tests on the specified interface.

Test 1: The evaluator exercised the TLS interface by making multiple connection attempts while in each presenting connection a certificate was presented with a different condition which should cause the certificate to be



considered invalid. The conditions attempted were: establishing a certificate path in which one of the issuing certificates is not a CA certificate, a missing basic constraint extension, a CA flag being false, a missing KeyCertSign key usage, and an incorrect path length. All connection attempts were rejected by the TOE. The evaluator also performed a valid connection using a valid certificate that chained to a trusted root which was successful, and a second connection using a certificate that did not chain to a trusted root which was an unsuccessful connection.

Test 2: The evaluator alternately configured a test server to send an authentication certificate 1) that is valid and, 2) that is expired. In each case, the evaluator then attempted connections from a test server to the TOE Cortex XSOAR Engine (client) and observed that the connections only succeeded if there were no expired certificates.

Test 3: The TOE supports OCSP for revocation. The evaluator attempted several connections to demonstrate that revocation checking of certificate in the chain was being performed. The evaluator provided a valid chain of certificates which allowed the connection to be established. The evaluator next provided chains where certificate in the path were revoked. The TOE rejected all connection attempts when a certificate in the chain was revoked.

Test 4: The TOE does not support CRL, only OCSP for revocation checking. A test server was configured with a certificate chain such that the subca's OCSP response was signed by a certificate that lacks the OCSPSigning purpose. The TOE correctly detects the invalid OCSP responder's certificate and immediately rejects the connection attempt.

Test 5: The evaluator alternately configured a test server to send an authentication certificate 1) that is valid, 2) that has one byte in the ASN1 field changed, 3) that has one byte in the certificate signature changed, and 4) that has one byte in the certificate public key changed. In each case, the evaluator then attempted connections from a test server to the TOE Cortex XSOAR Engine (client) and observed that the connections only succeeded if the certificate is not modified/corrupted.

Test 6: This test has been performed in Test 5 above.

Test 7: This test has been performed in Test 5 above.

Test 8: The TOE supports EC certificates and the evaluator attempted connections with the TOE using a valid certificate chain with certificates using a named curve and observed the connections succeed. The evaluator also attempted connections using a certificate chain with a certificate containing an explicit curve and observed that the connection was rejected by the TOE.

Test 9: This test has been performed in Test 8 above.

2.3.1.2 ASPP14:FIA_X509_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: 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 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.

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 evaluator alternately configured a test server to send an authentication certificate issued by a Sub CA with no BasicConstraints and with BasicConstraints but the CA Flag set to false. In each case, the evaluator then attempted to connect the TOE Cortex XSOAR Engine (as the TLSC TOE client) and the connection was rejected in each case.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.3.2 X.509 CERTIFICATE AUTHENTICATION (ASPP14:FIA_X509_EXT.2)

2.3.2.1 ASPP14:FIA_X509_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.2.2 ASPP14:FIA_X509_EXT.2.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined



Component TSS Assurance Activities: The evaluator shall check the TSS to ensure that it 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. 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. 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.

Section 6.2 of the ST states that certificates that are presented to the TOE must meet the x509v3 requirements as defined in RFC 5280 for TLS. If there are any issues with the certificate presented (as noted above in ASPP14:FIA_X509_EXT.1), the TOE will not accept the certificate and reject the connection. A log message will be generated, and an administrator will be required to address the problem noted for the connection to succeed.

The TOE implements the ability to perform certificate path validation on the certificate chain that is presented to it by the peer. The certificate path validation begins with the identity certificate presented by the peer, and then proceeds in checking the intermediate CA certificate(s) until it reaches the trusted CA certificate stored in the platform OS trust store. If no trusted CA certificate is found (i.e., the TOE cannot choose to use any certificate for path validation), the session will terminate.

The TOE also supports the revocation checking of the certificate presented using OCSP as specified in RFC 2560. In the event that the certificate is revoked following a check of its status, the TOE will terminate the connection. If the OCSP responder can't be reached, the administrator can configure if the connection is established or terminated prior to the event occurring. The 'TLS/X509 Configuration' section in the [Admin Guide] includes instructions on how the administrator can specify the default action in terms of the behavior of the TOE when a connection cannot be established during the validity check of a certificate used in establishing a trusted channel.

No configuration is needed for the operational environment.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: 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.

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.



The evaluator alternately configured a test server to send an authentication certificate with valid/accessible revocation servers and an authentication certificate with revocation information referring to an inaccessible revocation server. In each case, the evaluator then attempted to connect the TOE Cortex XSOAR Engine to the test server and observed that the connection was successful when the revocation server is accessible and failed when the revocation server is not accessible.

The evaluator then repeated this same test again after reconfiguring the TOE to allow the connection if the certificate revocation status cannot be determined and confirmed that the TOE accepted the connection.

Test 2: See Test Case FIA_X509_EXT.1-t1 where invalid certificates are tested. Note, that the TOE does not check revocation unless a valid certificate is presented. If an invalid certificate is presented, the TOE will immediately reject the connection without reaching out to the revocation server to validate the certificate. This is shown in the evidence under the FIA_X509_EXT.1 Test 1 results.

2.4 SECURITY MANAGEMENT (FMT)

2.4.1 SECURE BY DEFAULT CONFIGURATION (ASPP14:FMT_CFG_EXT.1)

2.4.1.1 ASPP14:FMT_CFG_EXT.1.1

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

Section 6.4 of the ST states the TOE is shipped with no default credential. The TOE relies on the operating system to enforce I&A and provide configurations for privileged users.

Guidance Assurance Activities: None Defined

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

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

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

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

The TOE does not use any default credentials; therefore, these tests are not applicable.



2.4.1.2 ASPP14:FMT_CFG_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

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

Platforms: Android....

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

Platforms: Microsoft Windows....

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

Platforms: Apple iOS....

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

Platforms: Linux....

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

Platforms: Oracle Solaris....

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

Platforms: Apple macOS....

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

The evaluator issued the `'ls -alR|grep -E '^.....w.'` and the `'find -L . -perm /002'` commands in order to check for any world writable files in the TOE's data directory. The evaluator verified that there are no world writable files in the TOE's data directory.



Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.4.2 SUPPORTED CONFIGURATION MECHANISM - PER TD0624 (ASPP14:FMT_MEC_EXT.1)

2.4.2.1 ASPP14:FMT_MEC_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

Section 6.4 of the ST states that the TOE stores its' configuration settings and logs in Linux recommended directories. Section "TOE Operations" in the [Admin Guide] states that the user must enable the following settings in the TOE application:

"Configure certification validation check"

"Configure revocation checking"

"Enable and configure the SAN trust list"

As stated by the ST, all of these options are stored on the TOE platform. Section "TLS/X509 Configuration" in the [Admin Guide] shows how to configure each one of these settings respectively.

Component Guidance Assurance Activities: None Defined

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

**Platforms: Android....**

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

Platforms: Microsoft Windows....

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

Platforms: Apple iOS....

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

Platforms: Linux....

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

Platforms: Oracle Solaris....

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

Platforms: Apple macOS....

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



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

The evaluator noted the file directories in which the TOE stores its configuration data. Using the utility Strace, the evaluator successfully verified that the TOE modifies the correct files in the correct locations after making changes to the TOE's configuration.

2.4.3 SPECIFICATION OF MANAGEMENT FUNCTIONS (ASPP14:FMT_SMF.1)

2.4.3.1 ASPP14:FMT_SMF.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

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

Section "TOE Operations" in the [Admin Guide] provides a sub-section of every management function claimed in the ST and supported by the TOE. The [Admin Guide] also provides the information required to perform the management duties associated with each management function.

Component Testing Assurance Activities: The evaluator shall test the application's ability to provide the management functions by configuring the application and testing each option selected from above. The evaluator is expected to test these functions in all the ways in which the ST and guidance documentation state the configuration can be managed.

The evaluator exercised each of the management functions claimed in the ST and verified the application's ability to provide the management functions by configuring the application and testing each of the selected options. The evaluator followed the instructions in the guidance documentation to test each of the claimed management functions accordingly.

2.5 PRIVACY (FPR)

2.5.1 USER CONSENT FOR TRANSMISSION OF PERSONALLY IDENTIFIABLE (ASPP14:FPR_ANO_EXT.1)



2.5.1.1 ASPP14:FPR_ANO_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.5 of the ST states the TOE does not transmit or store any personally identifiable information about an individual.

Component Guidance Assurance Activities: None Defined

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

This test is not applicable as the TOE does not collect or transmit any PII.

2.6 PROTECTION OF THE TSF (FPT)

2.6.1 ANTI-EXPLOITATION CAPABILITIES (ASPP14:FPT_AEX_EXT.1)

2.6.1.1 ASPP14:FPT_AEX_EXT.1.1

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

Section 6.6 of the ST states the TOE automatically enables ASLR when the application is compiled (with -pie build flag), and stack-based buffer overflow protection is enabled by default.

Guidance Assurance Activities: None Defined

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



Platforms: Android....

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

Platforms: Microsoft Windows....

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

Platforms: Apple iOS....

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

Platforms: Linux....

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

Platforms: Oracle Solaris....

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

Platforms: Apple macOS....

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

The evaluator took a process ID map using pmap from one instance of the TOE. The evaluator then installed the application on another instance of RHEL. The evaluator then took another process ID map using pmap. The evaluator compared the two and determined that the TOE does not share any memory mapping locations.

2.6.1.2 ASPP14:FPT_AEX_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

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



Platforms: Android....

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

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

Platforms: Microsoft Windows....

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

Platforms: Apple iOS....

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

Platforms: Linux....

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

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

Platforms: Oracle Solaris....

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

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

Platforms: Apple macOS....

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

The evaluator unpacked the TOE's rpm (application installation file). The evaluator then performed a grep search for all instances of mmap, mprotect, PROT_WRITE, and PROT_EXEC. The evaluator found some instances of mmap but insured that none of these instances are invoked with either 'PROT_WRITE' or 'PROT_EXEC'. Similarly, the evaluator found some instances of mprotect but insured that none of these instances are invoked with 'PROT_EXEC'. The evaluator did not find any instances of PROT_WRITE, or PROT_EXEC.

2.6.1.3 ASPP14:FPT_AEX_EXT.1.3



TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

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

Platforms: Android....

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

Platforms: Microsoft Windows....

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

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

Platforms: Apple iOS....

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

Platforms: Linux....

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

Platforms: Oracle Solaris....

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

Platforms: Apple macOS....

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



The evaluator checked the SELinux status on the Linux platform hosting the TOE and ensured that it was enabled by default and in enforce mode. With SELinux enabled and in enforce mode, the evaluator confirmed that the TOE was still able to operate normally.

2.6.1.4 ASPP14:FPT_AEX_EXT.1.4

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

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

Platforms: Android....

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

Platforms: Microsoft Windows....

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

Platforms: Apple iOS....

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

Platforms: Linux....

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

Platforms: Oracle Solaris....

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

Platforms: Apple macOS....



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

The evaluator captured a listing of the files before and after normal usage of the application. The evaluator executed the TOE and used it to modify the configuration file of the Engine (mimicking normal usage of the application). Comparison of the two output files shows that the TOE application writes data to /var/lib/demisto/. The evaluator verified that the TOE application does not write data to the executable directory.

2.6.1.5 ASPP14:FPT_AEX_EXT.1.5

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

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

Platforms: Microsoft Windows....

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

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

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

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

Tools such as Canary Detector may help automate these activities.

The evaluator unpacked the TOE's rpm (application installation file) to search for instances of the `stack_chk_guard` and `stack_chk_fail` keywords. The evaluator found that the stack protections exist.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined



2.6.2 USE OF SUPPORTED SERVICES AND APIs (ASPP14:FPT_API_EXT.1)

2.6.2.1 ASPP14:FPT_API_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

The vendor had submitted a separate proprietary document that lists all the Linux APIs used by the TOE.

Component Guidance Assurance Activities: None Defined

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

The evaluator compared the list of the TOE's claimed APIs with the list of supported APIs on each respective platform and ensured that all APIs listed in the TSS are supported.

2.6.3 SOFTWARE IDENTIFICATION AND VERSIONS (ASPP14:FPT_IDV_EXT.1)

2.6.3.1 ASPP14:FPT_IDV_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.6 of the ST explains Palo Alto Networks provides a version control system for its software components. The TOE has a unique software versioning that identifies major versions and their subsequent maintenance releases in the following form: <Major>.<Minor>.<Maintenance release> Build version.

Major and minor releases introduce new major and minor features for the product, and additional maintenance releases (build version) are released on a regular cadence to fix security issues or bugs identified with the release.

Component Guidance Assurance Activities: None Defined



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

The evaluator installed the TOE application and verified that the TOE’s versioning information can be found through its’ audit logs.

2.6.4 USE OF THIRD PARTY LIBRARIES (ASPP14:FPT_LIB_EXT.1)

2.6.4.1 ASPP14:FPT_LIB_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

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

The TOE application statically links only the intended libraries (BoringCrypto and Golang) into the TOE’s executable. As such, the TOE’s library directories do not contain any dynamic libraries except standard Linux kernel and c system libraries which are part of the Linux platform. They come standard as part of any Linux distribution.

The evaluator installed the application and navigated to its’ installation directory. The evaluator then used the Linux ldd command to list all the shared object dependencies. The evaluator then verified that the only libraries found are standard Linux kernel and c system libraries which are not third-party libraries.

2.6.5 INTEGRITY FOR INSTALLATION AND UPDATE (ASPP14:FPT_TUD_EXT.1)

2.6.5.1 ASPP14:FPT_TUD_EXT.1.1

TSS Assurance Activities: None Defined

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

Section “Software Download and Installation” in the [Admin Guide] includes instructions on how to install the TOE application using the required download link provided by Palo Alto. In case of an update, a download link must be



provided by Palo Alto and the same instructions can be followed to install the newer version of the TOE application.

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

The evaluator followed the guidance documentation to check for a new version of the TOE on Palo Alto's website. The evaluator observed that the version on the website matched the current running version of the TOE, therefore an update was not needed.

2.6.5.2 ASPP14:FPT_TUD_EXT.1.2

TSS Assurance Activities: None Defined

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

Section "Viewing the Current TOE Version" in the [Admin Guide] provides the procedures for querying the current version of the application. The version can be viewed through the TOE's audit logs.

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

The evaluator was able to view the TOE's current version through its' audit logs.

2.6.5.3 ASPP14:FPT_TUD_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

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

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

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

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



The evaluator recorded the hash of the TOE's executable in the TOE's installation directory and then proceeded to use the TOE normally. Afterwards, the evaluator queried the hash for the executable again. The evaluator confirmed that the hashes remained the same.

2.6.5.4 ASPP14:FPT_TUD_EXT.1.4

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

Section 6.6 of the ST states the TOE has specific versions, which can be queried by the user via the TOE's audit logs. New versions of the TOE are created by Palo Alto Networks, which an administrator can retrieve to update the current version of the TOE. During the installation process, a digital signature verification check can be performed by the platform to verify that the signed installer has not been modified. All versions of the installer are digitally signed by Palo Alto Networks using RSA 2048+ with SHA-256.

Customer Support will send verified customers an email with a download link. They also send license key and instructions in the email. The link is only valid for a limited number of downloads and time. The image is securely retrieved from the download server (<https://download.demisto.com>) The TOE cannot update its own binary code – it relies on the administrator to download and install the new version available. Based on the Linux distribution, the package can be RPM (e.g., Redhat) or DEB (e.g., Ubuntu).

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.6.5.5 ASPP14:FPT_TUD_EXT.1.5

TSS Assurance Activities: 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.

Section 6.6 of the ST states that the application is distributed based on the Linux distribution, the package can be RPM (e.g., Redhat) or DEB (e.g., Ubuntu).

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined



2.6.6 INTEGRITY FOR INSTALLATION AND UPDATE - PER TD0628 (ASPP14:FPT_TUD_EXT.2)

2.6.6.1 ASPP14:FPT_TUD_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: FPT_TUD_EXT.2.1: 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 correct format. This varies per platform:

Platforms: Android....

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

Platforms: Microsoft Windows....

The evaluator shall ensure that the application is packaged in the standard Windows Installer (.MSI) format, the Windows Application Software (.EXE) format signed using the Microsoft Authenticode process, or the Windows Universal Application package (.APPX) format. See

[https://msdn.microsoft.com/enus/library/ms537364\(v=vs.85\).aspx](https://msdn.microsoft.com/enus/library/ms537364(v=vs.85).aspx) for details regarding Authenticode signing.

Platforms: Apple iOS....

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

Platforms: Linux....

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

Platforms: Oracle Solaris....

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

Platforms: Apple macOS....

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



The evaluator successfully verified that the TOE installer is a Package Manager (.rpm) packaged file.

2.6.6.2 ASPP14:FPT_TUD_EXT.2.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: Platforms: Android....

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

Platforms: 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: Apple iOS....

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

Platforms:Linux...

The evaluator shall 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:Oracle Solaris...

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:Apple macOS...



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.

This assurance activity has been tested in FPT_TUD_EXT.1.3 where the evaluator shows evidence that the TOE does not modify its' own executables.

2.6.6.3 ASPP14:FPT_TUD_EXT.2.3

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

Section 6.6 of the ST states the TOE has specific versions, which can be queried by the user via the TOE's audit logs. New versions of the TOE are created by Palo Alto Networks, which an administrator can retrieve to update the current version of the TOE. During the installation process, a digital signature verification check can be performed by the platform to verify that the signed installer has not been modified. All versions of the installer are digitally signed by Palo Alto Networks using RSA 2048 with SHA-256.

Customer Support will send verified customers an email with a download link. They also send license key and instructions in the email. The link is only valid for a limited number of downloads and time. The image is securely retrieved from the download server (<https://download.demisto.com>) The TOE cannot update its own binary code – it relies on the administrator to download and install the new version available. Based on the Linux distribution, the package can be RPM (e.g., Redhat) or DEB (e.g., Ubuntu).

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.7 TRUSTED PATH/CHANNELS (FTP)

2.7.1 PROTECTION OF DATA IN TRANSIT (ASPP14:FTP_DIT_EXT.1)

2.7.1.1 ASPP14:FTP_DIT_EXT.1.1

TSS Assurance Activities: None Defined



Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

The ST does not claim platform provided functionality.

Component Guidance Assurance Activities: None Defined

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

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

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

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

Platforms: Android....

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

Platforms: Apple iOS....

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

Test 1: The TOE also encrypts its' traffic with TLS when communicating with its' Server. The evaluator established a connection between the TOE and its' Server. The result is that the traffic is encrypted in TLS.



Test 2: Using the packet capture for the evidence in test 1, the evaluator verified that there is no plaintext data in the network communication between the TOE and its' Server.

Test 3: The evaluator used Wireshark to open up the packet capture taken from test 1 and searched for the plaintext string "Admin123", which is the password for the admin account which was used to login to the TOE during this TLS connection captured in the referenced packet capture. The result is that the string does not show up in the packet capture. The evaluator repeated the same procedure with a hex editor.



3. PROTECTION PROFILE SAR ASSURANCE ACTIVITIES

The following sections address assurance activities specifically defined in the claimed Protection Profile that correspond with Security Assurance Requirements.

3.1 DEVELOPMENT (ADV)

3.1.1 BASIC FUNCTIONAL SPECIFICATION (ADV_FSP.1)

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

3.2 GUIDANCE DOCUMENTS (AGD)

3.2.1 OPERATIONAL USER GUIDANCE (AGD_OPE.1)

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

All necessary instructions to configure and use the TOE cryptography and initiate and verify TOE updates are provided by the TOE guidance documentation and have been identified throughout this AAR in the related assurance activities.

The 'Verify FIPS-CC Mode' section in the [Admin Guide] states that the FIPS-CC mode cannot be disabled, it requires all power-up self-tests to pass. If any of the self-tests fails, the system will not start up (i.e., in FIPS error state). If this occurs, uninstalling and re-installing the application is required. The FIPS-CC mode refers to a set of



features which meet the requirements for this evaluation and ensures that the TOE installation will operate in a manner consistent with the CC evaluation. This section in the [Admin Guide] also states that the TOE uses the BoringCrypto cryptographic module which uses NIST-validated and Approved algorithms as specified in #A2517. The cryptographic engine cannot be changed by the security administrator or user.

The "Update the TOE Version" section in [Admin Guide] describes how to use the update feature to upgrade, configure, and manage software versions. This includes instructions for installing and initiating the update process.

The 'Scope of Evaluation' section in the [Admin Guide] provides a detailed description of the scope of the CC evaluation. It states that the scope of evaluation only covers security functionalities specified by Protection Profile for Application Software and Functional Package for Transport Layer Security. The security functions tested are outlined in the Security Target (ST), and these functions include TLS trusted channels, X509 authentication, certificate validation and signature checking.

The various sections from the [Admin Guide] mentioned above provide instructions to ensure that the set of features which meet the requirements for this evaluation are enabled and that the TOE installation will operate in a manner consistent with the CC evaluation. There are also notes provided in the [Admin Guide] to clarify when a described feature is not included in the evaluation.

3.2.2 PREPARATIVE PROCEDURES (AGD_PRE.1)

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

Detailed step-by-step instructions for installing the TOE are provided by the [Admin Guide] referenced throughout the preceding Guidance Assurance Activities. The Guide addresses the claimed Linux platform.

3.3 LIFE-CYCLE SUPPORT (ALC)

3.3.1 LABELLING OF THE TOE (ALC_CMC.1)

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

The evaluator verified that the ST, TOE and Guidance are all labeled with identifiable versions. The evaluator checked the TOE version during testing by examining the actual devices used for testing.

3.3.2 TOE CM COVERAGE (ALC_CMS.1)



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

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

The evaluator verified that the ST, TOE and Guidance are all labeled with identifiable versions. The evaluator checked the TOE version during testing by examining the actual devices used for testing.

3.3.3 TIMELY SECURITY UPDATES (ALC_TSU_EXT.1)

Assurance Activities: The evaluator shall verify that the TSS contains a description of the timely security update process used by the developer to create and deploy security updates. The evaluator shall verify that this description addresses the entire application.

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

The evaluator shall verify that the description of this mechanism includes a method for protecting the report either using a public key for encrypting email or a trusted channel for a website.

Section 2.2.2.6 of the ST contains a description of the timely security update process used by the developer to deploy security updates. This section in the ST states that the trusted updates (signed RPM package) are securely delivered over HTTPS website and verified using approved digital signature methods.



3.4 TESTS (ATE)

3.4.1 INDEPENDENT TESTING - CONFORMANCE (ATE_IND.1)

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

While it is not necessary to have one test case per test listed in an Assurance Activity, the evaluator must document in the test plan that each applicable testing requirement in the ST is covered. The test plan identifies the platforms to be tested, and for those platforms not included in the test plan but included in the ST, the test plan provides a justification for not testing the platforms. This justification must address the differences between the tested platforms and the untested platforms, and make an argument that the differences do not affect the testing to be performed. It is not sufficient to merely assert that the differences have no affect; rationale must be provided. If all platforms claimed in the ST are tested, then no rationale is necessary. The test plan describes the composition of each platform to be tested, and any setup that is necessary beyond what is contained in the AGD documentation. It should be noted that the evaluator is expected to follow the AGD documentation for installation and setup of each platform either as part of a test or as a standard pre-test condition. This may include special test drivers or tools. For each driver or tool, an argument (not just an assertion) should be provided that the driver or tool will not adversely affect the performance of the functionality by the TOE and its platform.

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

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

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

The following diagram depicts the evaluator's test environment:

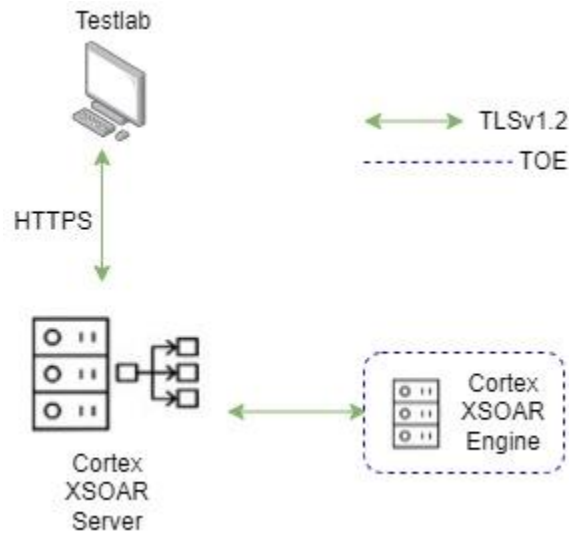


Figure 1 Test Setup

TOE Platforms:

- Cortex XSOAR Engine 6.6.0 running on RedHat 8.

Supporting Software:

- SSH Client – Putty version 0.74
- Wireshark version 3.4
- Strace version 4.11
- Nmap version 7.01 (Linux)
- Nmap version 7.92 (Windows)
- HxD version 2.0
- OpenSSL 1.0.2g
- Rpm2cpio (Linux RPM unpacking tool)

3.5 VULNERABILITY ASSESSMENT (AVA)

3.5.1 VULNERABILITY SURVEY (AVA_VAN.1)

Assurance Activities: The evaluator shall generate a report to document their findings with respect to this requirement. This report could physically be part of the overall test report mentioned in ATE_IND, or a separate document. The evaluator performs a search of public information to find vulnerabilities that have been found in similar applications with a particular focus on network protocols the application uses and document formats it parses. The evaluator shall also run a virus scanner with the most current virus definitions against the application files and verify that no files are flagged as malicious. The evaluator documents the sources consulted and the



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

The vulnerability analysis is in the Detailed Test Report (DTR) prepared by the evaluator. The vulnerability analysis includes a public search for vulnerabilities. The public search for vulnerabilities did not uncover any residual vulnerabilities.

An initial search was conducted on May 4, 2022, and a follow-up search was conducted on August 4, 2022 and then again on September 27, 2022. At that time all issues found were considered and a resolution determined for each issue. The results of that analysis are shown below. Updates to this public search were also performed. During an update the evaluator searched each site, for each term, but considered only new or changed issues that were returned by the public sites which appeared since the previous search. For the update, every issue was considered and a resolution was determined for each issue. The evaluator determined that there are no open public issues.

The evaluator searched the National Vulnerability Database (NVD) (<https://web.nvd.nist.gov/view/vuln/search>) and Vulnerability Notes Database (VND) (<http://www.kb.cert.org/vuls/>) with the following search terms:

- Palo Alto Networks
- Cortex XSOAR
- Cortex XSOAR Server
- Cortex XSOAR Engine
- Redhat Enterprise Linux v8.4
- Golang 1.16
- BoringCrypto

Additionally, the evaluator used the Windows Defender Antivirus app, checked to ensure the virus definitions were current, and ran a scan on the TOE Cortex XSOAR Engine. No issues were identified.