www.GossamerSec.com

Assurance Activity Report for TheJoin, Inc., Join-Virtual Mobile Platform 6.1.0

Version 0.3
02/08/2024

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

Document: AAR-VID11450

Revision History

Revision	Date	Authors	Summary
Version 0.1	01/04/203	Rizheng Sun	Initial draft
Version 0.2	01/26/203	Gossamer	Addressed ECR comments
Version 0.3	02/08/203	Gossamer	Addressed ECR comments

The TOE Evaluation was Sponsored by:

TheJoin, Inc.
3F,63, Bongeunsa-ro 30-gil
Gangnam-gu, Seoul, Republic of Korea

Evaluation Personnel:

- Raymond Smoley
- Rizheng Sun

Common Criteria Versions:

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

Common Evaluation Methodology Versions:

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

Table of Contents

1. Introduction

This document presents evaluations results of the TheJoin Inc., Join Virtual Mobile Platform 6.0 ASPP14/PKGTLS11 evaluation. This document contains a description of the assurance activities and associated results as performed by the evaluators.

1.1 Test Platforms

The TOE was tested on the following mobile devices.

Device Name	Processor	Operating System
Galaxy S22 Ultra 5G	Qualcomm Snapdragon 8 Gen 1 Mobile Platform	Android 13
Apple iPhone X	Apple A11 Bionic	Apple iOS 16

Table 1 Tested Devices

The following devices are being claimed as equivalent to the tested devices. The TOE runs on all the Samsung (VID11342, 04/26/2023 and VID11410, 10/23/2023) and Google (VID11317, 01/24/2023) devices listed below running Android 13. The TOE also runs on Apple iOS 16 (VID11349, 10/10/2023) on iPhone devices below. The same application runs on all Android devices and the same application runs on all iPhone devices.

Device Name	Operating System
Galaxy S23 Ultra 5G	Android 13
Galaxy S22 5G	Android 13
Galaxy S21 Ultra 5G	Android 13
Galaxy S20+ 5G	Android 13
Galaxy Z Flip	Android 13
Galaxy XCover Pro	Android 13
Galaxy A53 5G	Android 13
Galaxy XCover6 Pro	Android 13
Galaxy Z Flip5 5G	Android 13
Galaxy A52 5G	Android 13
Galaxy A71 5G	Android 13
Galaxy Tab Active3	Android 13
Galaxy S23 FE	Android 13
Google Devices	
Google Pixel 7 Pro	Android 13
Google Pixel 7	Android 13
Google Pixel 6 Pro	Android 13
Google Pixel 6	Android 13

Google Pixel 6a	Android 13
Google Pixel 5a-5G	Android 13
Google Pixel 5	Android 13
Google Pixel 4a-5G	Android 13
Google Pixel 4a	Android 13
Apple Devices	
iPhone 14 Plus	iOS 16
iPhone 14 Pro Max	iOS 16
iPhone 14 Pro	iOS 16
iPhone 14	iOS 16
iPhone SE (3rd gen)	iOS 16
iPhone 13 mini	iOS 16
iPhone 13 Pro Max	iOS 16
iPhone 13 Pro	iOS 16
iPhone 13	iOS 16
iPhone 12 mini	iOS 16
iPhone 12 Pro Max	iOS 16
iPhone 12 Pro	iOS 16
iPhone 12	iOS 16
iPhone SE (2nd gen)	iOS 16
iPhone 11 Pro Max	iOS 16
iPhone 11 Pro	iOS 16
iPhone 11	iOS 16
iPhone XS	iOS 16
iPhone XS Max	iOS 16
iPhone XR	iOS 16
iPhone 8 Plus	iOS 16
iPhone 8	iOS 16

Table 2 Equivalent Devices

The TOE was fully tested on two devices, a Samsung Galaxy S22 Ultra running Android 13 and an Apple iPhone X running iOS 16. The TOE is a standard Android/Apple application and can be installed on any of the evaluated devices listed while using the same TOE builds provided for testing. The TOE uses standard Android/iOS API calls and does not make direct hardware calls.

1.2 CAVP Certificate Mapping

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

Functions	Requirement	Cert #
Encryption/Decryption		
AES CBC (128 bits) AES GCM (128 bits)	ASPP14:FCS_COP.1/SKC	A3593
Cryptographic hashing		
SHA-256	ASPP14:FCS_COP.1/Hash	A3593
Keyed Hash		
HMAC-SHA256	ASPP14:FCS_COP.1/KeyedHash	A3593
Digital Signature		
RSA Sign/Verify 2048 bits	ASPP14:FCS_COP.1/Sig	A3593

2. Protection Profile SFR Assurance Activities

This section of the AAR identifies each of the assurance activities included in the claimed Protection Profile and Extended Packages. This section also describes the findings for each activity.

2.1 Cryptographic support (FCS)

2.1.1 Cryptographic Key Establishment (ASPP14:FCS_CKM.2)

2.1.1.1 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/AK. If the ST specifies more than one scheme, the evaluator shall examine the TSS to verify that it identifies the usage for each scheme. (TD0717 applied)

Section 6.1 of the ST states the TOE supports RSA key establishment (key size 2048) as part of HTTPS/TLS. The TOE acts as a client for TLS or HTTPS (RSA) when communicating with the VMI Server. Key generation is not required as the ASPP14 notes that if the TOE acts as a receiver in the RSA key establishment scheme, the TOE does not need to implement RSA key generation. That is the reason FCS_CKM.1/AK is not included in the ST.

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).

Section 7 of the User Guide states that the TOE is in the evaluated configuration by default and that no configuration is needed for the evaluated cryptography to be used.

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

Key Establishment Schemes

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

SP800-56A Key Establishment Schemes

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

Function Test

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

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

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

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

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

Validity Test

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

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

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

SP800-56B Key Establishment Schemes

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

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

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

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

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

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

RSA-based key establishment

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

Diffie-Hellman Group 14

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

FFC Schemes using 'safe-prime' groups

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

The TOE's RSA key exchange mechanism is used in the TLS handshake process and during both product development and evaluation testing, the TOE's implementation undergoes testing to ensure TLS compatibility. Any defect in the RSA key exchange mechanism would result in an inability to negotiate the TLS_RSA_WITH_AES_128_CBC_SHA256 ciphersuite with a separate, known good implementation. Gossamer's TLSC testing results demonstrate the TOE conforms to the specifications.

2.1.2 Cryptographic Key Generation Services (ASPP14:FCS_CKM_EXT.1)

2.1.2.1 ASPP14:FCS_CKM_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 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 TOE acts as a client for TLS when communicating with the VMI Server and does not generate asymmetric cryptographic keys.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

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

2.1.3.1 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 the TOE uses hash functions that include SHA-256 as defined in FIPS 180-4. The hash functions are used as part of HTTPS/TLS cipher negotiation.

Component Guidance Assurance Activities: None Defined

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

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

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

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

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

Test 4: Selected Long Messages Test - Byte oriented Mode The evaluators devise an input set consisting of m/8 messages, where m is the block length of the hash algorithm. The length of the ith message is 512 + 8*99*i, where 1 <= i <= m/8. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

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

See Section 1.2 for a listing of applicable CAVP certificates.

2.1.4 Cryptographic Operation - Keyed-Hash Message Authentication - per TD0717 (ASPP14:FCS_COP.1/KeyedHash)

2.1.4.1 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: The evaluator shall perform the following activities based on the selections in the ST.

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.

See Section 1.2 for a listing of applicable CAVP certificates.

2.1.5 Cryptographic Operation - Signing - per TD0717 (ASPP14:FCS_COP.1/Sig)

2.1.5.1 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.

See Section 1.2 for a listing of applicable CAVP certificates.

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

2.1.6.1 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 TOE negotiates the claimed ciphers by default. Section 7 of the User Guide states that the TOE is in the evaluated configuration by default and Section 12.1 of the User Guide lists TLS_RSA_WITH_AES_128_GCM_SHA256 as the evaluated cipher.

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

AES-CBC Known Answer Tests

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

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

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

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

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

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

AES-CBC Multi-Block Message Test

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

AES-CBC Monte Carlo Tests

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

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

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

AES-GCM Monte Carlo Tests

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

128 bit and 256 bit keys

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

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

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

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

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

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

AES-XTS Tests

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

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

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

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

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

AES-CCM Tests

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

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

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

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

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

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

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

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

Variable Associated Data Test

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

Variable Payload Test

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

Variable Nonce Test

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

Variable Tag Test

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

Decryption-Verification Process Test

To test the decryption-verification functionality of AES-CCM, for each combination of supported associated data length, payload length, nonce length, and tag length, the evaluator shall supply a key value and 15 sets of input plus ciphertext, and obtain the decrypted payload. Ten of the 15 input sets supplied should fail verification and five should pass.

AES-CTR Tests

Test 1: Known Answer Tests (KATs)

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

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

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

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

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

Test 2: Multi-Block Message Test

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

Test 3: Monte-Carlo Test

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

The evaluator shall test the encrypt functionality using 200 plaintext/key pairs. 100 of these shall use 128 bit keys, and 100 of these shall use 256 bit keys. The plaintext values shall be 128-bit blocks. For each pair, 1000 iterations shall be run as follows:

For AES-ECB mode

```
# Input: PT, Key

for i = 1 to 1000:

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

PT = CT[i]
```

The ciphertext computed in the 1000th iteration is the result for that trial. This result shall be compared to the result of running 1000 iterations with the same values using a known good implementation.

See Section 1.2 for a listing of applicable CAVP certificates.

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

2.1.7.1 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 6.1 of the ST states that the TOE invokes the evaluated platform provided functionality for its cryptographic functions.

Component Guidance Assurance Activities: None Defined

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

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

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

Platforms: Android....

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

Platforms: Microsoft Windows....

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

Platforms: Apple iOS....

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

Platforms: Linux....

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

Platforms: Oracle Solaris....

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

Platforms: Apple macOS....

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

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

The evaluator decompiled the TOE and found that the javax.crypto.KeyGenerator, /dev/random, and /dev/urandom are used for Android. SecRandomCopyBytes and /dev/random are used for iOS.

2.1.8 Storage of Credentials (ASPP14:FCS_STO_EXT.1)

2.1.8.1 ASPP14:FCS_STO_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.1 of the ST explains the client implements functionality to securely store server and account information in a local database. For the VMI server, the TOE stores the server address, port, username and password. The stored password is encrypted with AES128 CBC mode according to FCS_COP.1/SKC

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_EXT.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

For Android version of the application, the evaluator searched the decompiled version of the TOE using grep for the keyword “AES_CBC”. The evaluator found calls to AES_CBC encrypt and decrypt operations and concluded that the claimed encryption mode is supported.

For iOS version of the application, the evaluator searched the otool decompiled binary of the TOE using grep for “Sec” and "0x00000001" in order to find references to kSecClassKey, kSecAttrKeyTypeRSA, and SecItemAdd from the Apple CryptoKit framework. The evaluator concluded that the TOE is using these functions to store credentials in a keychain.

2.1.9 TLS Protocol (PKGTLS11:FCS_TLS_EXT.1)

2.1.9.1 PKGTLS11:FCS_TLS_EXT.1.1

TSS Assurance Activities: None Defined

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

Section 7 of the User Guide states no configuration is needed for evaluated cryptography to be used, and Section 12 of the User Guide states TLS_RSA_WITH_AES_128_GCM_SHA256 as the evaluated TLS cipher.

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.1.10 TLS Client Protocol (PKGTLS11:FCS_TLSC_EXT.1)

2.1.10.1 PKGTLS11: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 states the TOE communicates with the VMI Server using HTTPS/TLS. The HTTPS portion of the connection is implemented by the platform. The TOE supports protected communication channels using TLS v1.2 (RFC 5246) secure communication protocols. The TOE supports use of the following ciphersuite: TLS_RSA_WITH_AES_128_GCM_SHA256. This ciphersuite matches the requirement.

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.

Section 7 of the User Guide states no configuration is needed for evaluated cryptography to be used.

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.

These tests were performed for both the Android and iOS applications:

Test 1: The evaluator established a TLS session between the TOE and the test server for the claimed ciphersuite. The evaluator used a network sniffer to capture the TLS session negotiation and observed that the expected TLS cipher is negotiated.

Test 2: The evaluator configured the established a TLS session with the TOE and a test server. 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 with the TOE and a test server. A modified test server negotiates an RSA ciphersuite, but returns an ECDSA 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_NUL_NULL ciphersuite. The evaluator then attempted to establish a TLS session from the TOE 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 made connection attempts from the client to the test server. The server implementation of the TLS protocol was modified as stated in the 5 scenarios described by the Assurance Activity. The evaluator inspected each packet captures to ensure that the connections are rejected for each scenario. Note 5.3 and 5.5 were not performed as the TOE does not claim DHE or ECDHE ciphersuites.

2.1.10.2 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 of the ST states that the TOE the following reference identifiers are supported – optional Subject Alternate Name (SAN) (i.e., DNS, IP Address, or URI) and required Common Name (CN). The TOE supports wildcard reference identifiers in both the SAN and CN. If present, the TOE will check the correctness of the SAN extension, otherwise the TOE will check the CN field against the established reference identifier supplied by the user. Certificate pinning is 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.

Section 7 of the User Guide identifies the menu to set the server name or IP address and states this information is used for TLS certificate validation.

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.come) and verify that the connection fails.

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

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

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

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

These tests were performed for both the Android and iOS applications:

Test 1: The evaluator established a TLS session the TOE 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 from the TOE targeting a server using a server certificate that does not contain an identifier in the Subject Alternative Name (SAN) and a bad Common Name (CN). 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 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 attempted to connect to a server with a server certificate that contains a CN that matches the reference identifier and does not contain the SAN extension. The connection was accepted.

Test 4: The evaluator established a TLS session from the TOE 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: The evaluator configured a test server to use a server certificate containing a reference identifier as described by test 5 from the above assurance activity. The only acceptable combination was SAN=*.example.com .

Test 6: The evaluator configured a test server with a correct SAN URI and the connection was successful. The evaluator then configured a test server with an incorrect SAN URI and the connection was rejected.

Test 7: The TOE does not support certificate pinning

2.1.10.3 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 - the selection for override 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.

Test 1: This test has been performed in ASPP14:FIA_X509_EXT.1 test case 1.

Test 2: This test has been performed in ASPP14:FIA_X509_EXT.1 test case 3.

Test 3: This test has been performed in ASPP14:FIA_X509_EXT.1 test case 2.

Test 4: This test has been performed in PKGTLS11: FCS_TLSC_EXT.1.2.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.1.11 TLS Client Support for Renegotiation (PKGTLS11:FCS_TLSC_EXT.4)

2.1.11.1 PKGTLS11:FCS_TLSC_EXT.4.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

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

Test 1: The evaluator shall use a network packet analyzer/sniffer to capture the traffic between the two TLS endpoints. The evaluator shall verify that either the 'renegotiation_info' field or the SCSV cipher suite is included in the ClientHello message during the initial handshake.

Test 2: The evaluator shall verify the Client's handling of ServerHello messages received during the initial handshake that include the 'renegotiation_info' extension. The evaluator shall modify the length portion of this field in the ServerHello message to be non-zero and verify that the client sends a failure and terminates the connection. The evaluator shall verify that a properly formatted field results in a successful TLS connection.

Test 3: The evaluator shall verify that ServerHello messages received during secure renegotiation contain the 'renegotiation_info' extension. The evaluator shall modify either the 'client_verify_data' or 'server_verify_data' value and verify that the client terminates the connection.

These tests were performed for both the Android and iOS applications:

Test 1 – The evaluator configured the TOE to connect to a test server using TLS. The evaluator configured the server to connect normally with renegotiation. The evaluator observed the 'renegotiation_info' field in the handshake as expected.

Test 2 - The evaluator configured the TOE to connect to a test server using TLS. The evaluator configured the server to connect normally with renegotiation. The evaluator modified the length field of the ServerHello to be non-zero and the connection failed.

Test 3 - The evaluator configured the TOE to connect to a test server using TLS. The evaluator configured the server to connect normally with renegotiation. The evaluator modified server_verify_data field and the connection failed. The evaluator tested with an unmodified connection and the TOE connected as expected.

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 - per TD0756 (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 only sensitive data in the TOE is the server password and that is protected as described in FCS_STO_EXT.1. The evaluator cannot identify any other sensitive data from the descriptions in the TSS.

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.

If 'implement functionality to encrypt sensitive data as defined in the PP-Module for File Encryption' or 'protect sensitive data in accordance with FCS_STO_EXT.1' is selected, 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. (TD0756 applied)

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.

No test assurance activities are defined for this component because the only sensitive data is addressed in ASPP14:FCS_STO_EXT.1

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

2.2.2.1 ASPP14:FDP_DEC_EXT.1.1

TSS Assurance Activities: None Defined

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

Section 8 of the User Guide lists the resources needed by the application and a justification for why they are needed. The resources are listed for both Android and iOS.

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.

Android - On access, the TOE identifies and requests permissions. The TOE only prompts the user for access to “dangerous” permissions including access to Camera, Microphone, and Location Services. The other permissions including bluetooth, network connectivity, vibration, phone, and fingerprint hardware that are not considered dangerous are allowed upon download. This is consistent with the User Guide.

iOS - Upon normal use of the TOE, the app identifies and requests permissions for use of the Camera. Looking at the ‘J-VMP Client’ app in the settings on the phone, the TOE identifies use of the camera, Siri & Search, Notifications, and Background app refresh. The User Guide clarifies that for an iOS device, the TOE can access location services, camera, photos, microphone, notifications, and cellular data.

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.

The product does not need access to 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 WMAppManifest.xml file for a list of required capabilities. The evaluator shall identify the required information repositories when the application is first installed. This includes permissions such as ID_CAP_CONTACTS,ID_CAP_APPOINTMENTS,ID_CAP_MEDIALIB and so on. A complete list of Windows App permissions can be found at:

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

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

Platforms: Apple iOS....

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

Platforms: Linux....

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

Platforms: Oracle Solaris....

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

Platforms: Apple macOS....

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

The evaluator examined the application permissions of the TOE on access and found no claims for access to sensitive information repositories which is consistent with the selection in the ST.

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.

For both Android and iOS, the evaluator performed the following tests:

Test 1 - The evaluator sniffed the network to capture traffic while the TOE was running. The TOE only created network traffic when it connected to its server as expected.

Test 2- The evaluator performed a port scan and verified that the TOE does not listen on any ports. In the iOS case, two ports were open but the evaluator verified that the ports were unimpacted by the TOE.

2.3 Identification and authentication (FIA)

2.3.1 X.509 Certificate Validation - per TD0780 (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 TSS states the TOE performs certificate validation checking for TLS connections. The following fields are verified as appropriate: SAN checks, key usages, chain validation, and lastly expiration status. Wildcards are allowed in certificates. Both Android and iOS applications support OCSP when performing validity checks. Both applications do not accept certificates as valid when revocation status cannot be determined.

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 â€“ 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 TSF to reject certificates if it cannot access valid status information, if so configurable. Then the evaluator shall ensure the TSF has no other source of revocation information available and configure the OCSP server or use a man-in-the-middle tool to present an OCSP response signed by a certificate that does not have the OCSP signing purpose and which is the only source of revocation status information advertised by the CA issuing the certificate being validated. The evaluator shall verify that validation of the OCSP response fails and that the TOE treats the certificate being checked as invalid and rejects the connection. If CRL is selected, the evaluator shall likewise configure the CA to be the only source of revocation status information, and sign a CRL with a certificate that does not have the cRLsign key usage bit set. The evaluator shall verify that validation of the CRL fails and that the TOE treats the certificate being checked as invalid and rejects the connection. (TD0780 applied)

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.

These tests were performed for both the Android and iOS applications:

Test 1 :

Part a) For this test, the evaluator configured stunnel on a test server to send an authentication certificate that is valid. The evaluator then attempted to connect the TLSC TOE client to the test server and the connection was successful.

Part b) The evaluator ensured there was no root CA configured on the TOE, removing any root CA if necessary, and then attempted a connection. The connection was rejected as expected.

Part c) For this test, the evaluator alternately configured stunnel on a test server to send an authentication certificate with:

1) no BasicConstraints

2) BasicConstraints but the CA Flag set to false

3) an intermediate CA containing no keyCertSign purpose

4) an intermediate with a path length field set too low

In each case, the evaluator then attempted to connect the TLSC TOE client to the test server and the connection was rejected.

Test 2: The evaluator configured valid server and client certificates. A successful connection was made. The evaluator then configured a server certificate that was expired. The connection was refused. The evaluator then configured a server certificate that had an expired subCA. The connection was refused.

Test 3: The evaluator configured valid server and client certificates. A successful connection was made. The evaluator then configured a server certificate that is revoked with OCSP in accordance with RFC 6960. The connection was refused. The evaluator then configured a subca that is revoked with OCSP. The connection was refused.

Test 4: The evaluator alternately configured stunnel on a test peer to send an authentication certificate 1) that is valid, 2) that is issued by an intermediate CA whose issuer CA refers to an OCSP revocation server where the signer lacks OCSPSigning, 3) that is issued by an intermediate CA referring to an OCSP revocation server where the signer lacks OCSPSigning.

Only the valid authentication certificate was able to make a successful connection. The latter two connections were refused.

Test 5: The evaluator configured the 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.

The connection was successful only in the valid case, and was refused in each case with a modified certificate.

Test 6: This test was performed with test 5.

Test 7: This test was performed with test 5.

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.

These tests were performed for both the Android and iOS applications:

Test 1 - The evaluator configured a test server for TLS. The test server was configured with a certificate that did not contain the basicConstraints extension. A connection attempt was made and the connection was refused.

Test 2- The evaluator configured a test server for TLS. The test server was configured with a certificate that did not have the CA flag in the basicConstraints extension set.   A connection attempt was made and the connection was refused.

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 6.3 of the ST states the TOE comes pre-loaded with the certificate it uses. Both applications do not accept certificates as valid when revocation status cannot be determined.

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.

These tests were performed for both the Android and iOS applications:

Test 1 - The evaluator installed a certificate on the TOE with a valid certificate path and was able to make a network connection. The evaluator then blocked access to the OCSP server. The evaluator then attempted to connect to a web site. The TOE rejected the server certificate because it could not obtain a revocation status.

Test 2: See FIA_X509_EXT.1 test case 5 where it is demonstrated that a connection cannot be established if a revocation server cannot be reached.

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 does not include any predefined or default credentials.

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.

These tests are not applicable as the TOE does not have any default credentials. The evaluator demonstrated the TOE does allow for the ability to connect to an external VMI server using a set of credentials configured by the VMI server, however the TOE does revert to minimal functionality when not connected to the external server.

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.

Test - For Android, the evaluator ran ls -alR|grep -E '$....... (r|-w|--x)' inside the application's data directories to ensure that all files are not world-accessible (either read, write, or execute). The command did not print any files. The evaluator also verified that no sensitive data was written to external storage.

For iOS, the evaluator searched through the decompiled application used for FPT_AEX_EXT.1.1. The evaluator did not find any evidence of the File Data Protections classes (NSFIleProtectionComplete, NSFIleProtectionCompleteUnlessOpen, NSFIleProtectionCompleteUntilFirstUserAuthentication, and NSFIleProtectionNone). The evaluator then repeated this search for the iOS equivalent Keychain Data Protection classes (kSecAttrAccessibleWhenUnlocked, kSecAttrAccessibleAfterFirstUnlock, kSecAttrAccessibleAlways, kSecAttrAccessibleWhenPasscodeSetThisDeviceOnly) and found evidence of those classes being used. From this the evaluator concluded that no files were using an incorrect Data Protection Class.

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 TD0747 (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 the evaluated Android platform on which the TOE executes automatically uses /data/data/package/shared_prefs/ to store configuration options and settings. For an iOS platform, all settings are stored in the iOS the user defaults system.

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 inspect the TSS and verify that it describes what Android API is used (and provides a link to the documentation of the API) when storing configuration data. The evaluator shall run the application and verify that the behavior of the TOE is consistent with where and how the API documentation says the configuration data will be stored. (TD0747 applied)

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.

Section 6.4 of the ST states the evaluated Android platform on which the TOE executes automatically uses /data/data/package/shared_prefs/ to store configuration options and settings. For an iOS platform, all settings are stored in the iOS the UserDefaults system.

For Android, the evaluator ran the application and made security relevant changes. The Shared-Preferences xml file was updated as a result.

For iOS, the evaluator performed a static analysis on the decompiled .ipa file to ensure that the default key-value preference storage was being used. The evaluator searched for NSUserDefaults, the platform provided class used to access key-value storage. The evaluator also searched for its associated function, setObject, which is used to add entries to the storage. The evaluated noted that similar classes were reported near the occurrences of setObject to ensure that this was the correct setObject being called.

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 6.4 of the ST state the TOE provides the ability to specify the network address of a VMI server. The J-VMP Client can enable the Remember Password setting for each account. Section 7.1.1 of the User Guide shows the menu to set the network address. Section 7.1.3 of the User Guide shows the menu option for remembering the password.

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.

For both Android and iOS versions of the TOE, the network address can be found in the menus to configure server information that is used to make the initial connection to the J-VMP server. At any time, the evaluator was able to disconnect from the server and modify these settings. The set the remember password option was tested as part of CFG_EXT.1.1, test 1 and FMT_MEC_EXT.1.1, test 1.

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 TSS states the J-VMP client does not collect any PII and does not intentionally transmit any PII over a network.

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 transmit 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. If any explicitly-mapped exceptions are claimed, the evaluator shall check that the TSS identifies these exceptions, describes the static memory mapping that is used, and provides justification for why static memory mapping is appropriate in this case. (TD0798 applied)

Section 6.6 of the TSS states that the TOE libraries on Android are compiled with the '-fstack-protector-all -fno-exceptions' flags in order to enable ASLR and stack-based buffer over flow protections. On iOS the ASLR feature (-pie) is not set by a compiler flag, because it is on by default on the C-language compiler and this setting is required by the Apple App store.

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 except for any exceptions claimed in the SFR. For these exceptions, the evaluator shall verify that this analysis shows explicit mappings that are consistent with what is claimed in the TSS. 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. (TD0798 applied)

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.

For Android, the evaluator ran the app on the same device, uninstalling, rebooting, and reinstalling the TOE between collecting instances of the mappings.   The evaluator compared the results. The mappings were different between the devices.

For iOS, the evaluator performed a static analysis of the TOE. The evaluator was able to find several instances of the term mmap occur in the files. The evaluator was able to conclude that no arguments are provided that request a mapping at a fixed address.

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.

For both Android and iOS, the evaluator decompiled the application. After the decompilation, the evaluator performed a search for the mprotect permissions on the entire implementation of the application. The evaluator no instances of mprotect being used, and verified that mmap is not called with PROT_WRITE or PROT_EXEC in the Android application.

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.

For both Android and iOS, applications cannot disable security features, therefore this requirement is met and no evaluation activity is required.

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.

For Android, the evaluator used the TOE to make a connection and then examined the /data/data/package directory as instructed. The directory did not contain any executable files.

For iOS, no test is required.

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.

For both Android and iOS, the evaluator was able to decompile the applications and demonstrate the __stack_chk_fail symbol was present as required.

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.

Section 7 of ST provides a list of the platform APIs that are used by the TOE. The evaluator compared the list with the docs online to ensure each API was documented in the public. The evaluator relied on the https://developer.android.com/reference website for the Java API definitions. The evaluator used the https://developer.apple.com/documentation/ website for the iOS APIs.

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 states the platform user interface provides a method to query the current version of many components, including the TOE software. The TOE software version can be accessed on the Settings display in both versions of the application.

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.

For both Android and iOS, the evaluator demonstrated the version of the application is available to the user.

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 evaluator installed the application and searched for its dynamic libraries. In both Android and iOS, the evaluator only found evidence of the claimed libraries from the ST.

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 9 of the User Guide explains how to perform updates.

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 TOE uses the platform functionality to check for updates and patches to application software. The TOE is distributed on the iOS App Store and the Android Play Store and uses the associated repository to manage its versions. On each platform, the evaluator invoked the platform to check for updates. No updates were available at the time of testing.

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 10 of the User Guide identifies the menu for querying the current version of the application.

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 checked the version for both test devices using the system settings to display the reported information of the TOE.

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.

For Android, the evaluator recorded every file on the filesystem prior to installation of the application, and then installed and ran the application by connecting to the VMI Server. The evaluator then uninstalled the application and compared the resulting filesystem to the initial record. The only differences that were seen were related to things were expected to occur with normal use of TOE.

For iOS, the evaluator considered the requirement met because the platform forces applications to write all data within the application working directory (sandbox).

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 (J-VMP client) application is available through the Google Playstore and the Apple store. The platform will be providing all required capabilities for trusted updates for store versions.

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.

Section 5.1.6.5 of the ST identifies the method of distribution 'as an additional package'. The ST has included FPT_TUD_EXT.2

Guidance Assurance Activities: None Defined

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

See FPT_TUD_EXT.2

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 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 vendor provided the application in the form of an APK file for Android and an IPA file for iOS. These file formats were used for the entirety of testing. The evaluator examined these formats in depth in *Section 2.1 TOE Installation and Setup* where the evaluator decompiled each app and found that it followed the typical format for each. Additionally, each application is distributed using the platform application store which only provides platform-formatted applications.

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: Apple iOS....

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

All Other Platforms...

The evaluator shall record the path of every file on the entire filesystem prior to installation of the application, and then install and run the application. Afterwards, the evaluator shall then uninstall the application, and compare the resulting filesystem to the initial record to verify that no files, other than configuration, output, and audit/log files, have been added to the filesystem. (TD0664 applied)

The test requirements for this SFR are met by platform application sandbox requirements for both Android and iOS.

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 can be downloaded from the Google Playstore and the Apple store. Both repositories provide signed applications for installation.

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 - per TD0743 (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.

Section 6.7 of the ST states the TOE utilizes the platform API to establish HTTPS connections to a VMI server. The TOE utilizes its internal cryptographic library to establish TLS1.2 connections to VMI server.

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 - For both Android and iOS, the evaluator the evaluator performed a packet capture while using the TOE in the FDP_NET_EXT.1. The packet capture demonstrates the TOE establishes a connection with a server using TLS.

Test 2 - For both Android and iOS, the evaluator the evaluator performed a packet capture while using the TOE in the FDP_NET_EXT.1. The evaluator examined the bytes and could not find any bytes that contained human readable or sensitive data.

Test 3 - For both Android and iOS, the evaluator the evaluator performed a packet capture while using the TOE in the FDP_NET_EXT.1 where the evaluator captured network traffic of the TOE while connecting to a remote system for remote login. During this test, the evaluator provided credentials in the form of a username and password. The evaluator searched the evidence provided for this test and could not find the credentials present in the provided packet capture in a plaintext format.

3. Protection Profile SAR Assurance Activities

3.1 Development (ADV)

3.1.1 Basic Functional Specification (ADV_FSP.1)

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

The assurance activities from the ASPP14/PKGTLS11 have been performed and the analysis of the evaluator is documented in the previous sections of this document.

3.2 Guidance documents (AGD)

3.2.1 Operational User Guidance (AGD_OPE.1)

Assurance Activities: Some of the contents of the operational guidance will be verified by the assurance activities in Section 5.1 and evaluation of the TOE according to the [CEM]. The following additional information is also required. If cryptographic functions are provided by the TOE, the operational guidance shall contain instructions for configuring the cryptographic engine associated with the evaluated configuration of the TOE. It shall provide a warning to the administrator that use of other cryptographic engines was not evaluated nor tested during the CC evaluation of the TOE. The documentation must describe the process for verifying updates to the TOE by verifying a digital signature â€" 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.

- No user configuration is available for the cryptographic features of the TOE.
- Section 9 of the User Guide describes updates including the how to obtain them and install them.
- Section 6 explains the security features being evaluated.

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.

Section 6 of the User Guide identifies both the evaluated Android and iOS platforms in a table, and identifies the ST where matching evaluated platforms are identified in section 1.3.

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 User Guide identifies a specific version of the TOE for which it is relevant.

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 User Guide identifies a specific version of the TOE for which it is relevant and identifies the platforms for the evaluation.

3.3.3 Timely Security Updates (ALC_TSU_EXT.1)

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

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

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

Section 6.6 of the ST states that TheJoin accepts bug reports (including reports for security vulnerabilities) through email (bugs@thejoin.co.kr) and web (http://www.thejoin.co.kr) support channels. TheJoin reviews all bug reports when making product changes to resolve issues associated with the TOE. TheJoin makes updates and code patches to resolve issues as quickly as possible, and makes updates available to customers. TOE updates are distributed through the Apple App Store and Google Play. For maximum compatibility, TheJoin recommends customers use the iOS and Android update mechanisms to keep the J-VMP client up-to-date

3.4 Tests (ATE)

3.4.1 Independent Testing - Conformance (ATE_IND.1)

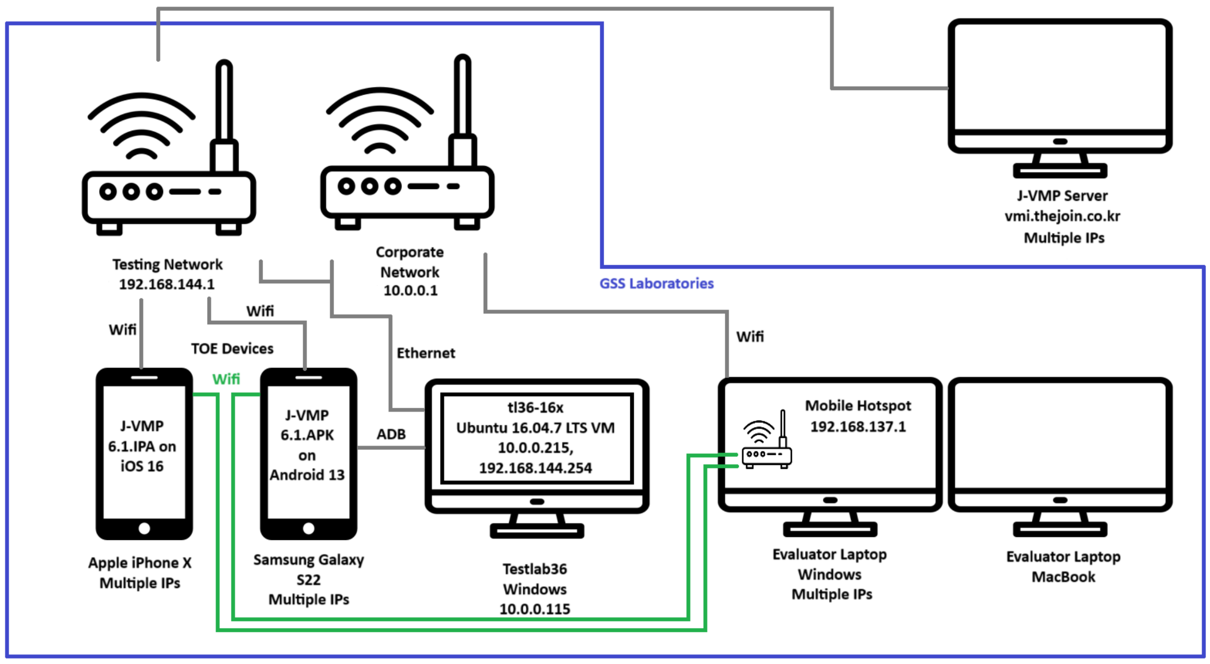
Assurance Activities: The evaluator shall prepare a test plan and report documenting the testing aspects of the system, including any application crashes during testing. The evaluator shall determine the root cause of any application crashes and include that information in the report. The test plan covers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

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

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

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

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



The evaluator used the following test tools:

- Wireshark Version 3.4.0
- OpenSSL version 1.0.1f
- Ubuntu Version 16.04
- Micro-httpd (comes with Ubuntu)
- Tcpdump (comes with Ubuntu)
- Adb (Android Debug Bridge) version 1.0.32
- Evaluator developed test scripts

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 evaluator searched the National Vulnerability Database (https://web.nvd.nist.gov/vuln/search), Vulnerability Notes Database (http://www.kb.cert.org/vuls/) on 1/24/2024 with the following search terms: "FBLPromises", "FirebaseCore", "FirebaseCoreInternal", "FirebaseMessaging", "FirebaseInstallations", "GoogleDataTransport", "GoogleUtilities", "nanopb", "vmi.thejoin.co.kr", "thejoin", "J-VMP", "Join-Virtual Mobile Platform", "SKInfosec", "VMI", "openSSL", "libvmi", "libchromium", "libconscrypt", "conscrypt", "google.conscrypt", "swift", "libswift", "skia", "AFNetworking", "ASIHTTPRequest", "Libegal", "Libjpeg", "LibOpenGLRender", "FMDB", "ffmpeg", "G726", "EGOImageLoading", "MBProgressHUD", "SFHFKeychainUtils", "Rechability", "SBJson", "SPLockScreen", "TheSideBarController", "JSBadgeView", "X264-152", "Com.google.code.gson", "Org.samba.jcifs", "Fr.avianey.com.viewpagerindicator", "Com.github.anzaizai:EasySwipeMenuLayout".

No residual vulnerabilities exist in the TOE.

The virus scanner test is not required as the TOE is an Android and iOS application – neither of which require the test.