www.GossamerSec.com

Assurance Activity Report for Cisco Jabber 15.2

Version 0.2
06/25/26

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-11635

Revision History

Revision	Date	Authors	Summary
Version 0.1	05/25/26	Kalmus	Initial draft
Version 0.2	06/25/26	Kalmus	Addressed ECR Comments

The TOE Evaluation was Sponsored by:

Cisco Systems, Inc.
170 West Tasman Dr.
San Jose, CA 95134

Evaluation Personnel:

- Douglas Kalmus

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 Cisco Jabber 15.2 ASPP14/PKGTLS11/VVoIPAS10 evaluation. This document contains a description of the assurance activities and associated results as performed by the evaluators.

1.1.1 References

The following evidence was used to complete the Assurance Activities:

- [ST] – Cisco Jabber 15.2 Security Target Version 0.5
- [AGD] – Cisco jabber 16.2 Common Criteria Configuration Guide Version 0.5

1.1.2 CAVP Certificate Equivalence

The TOE is the Cisco Jabber version 15.2. The TOE is a software-only client application that executes on a Windows 11 platform. For this evaluation, the Windows 11 device was a laptop with an Intel Core i5-1135G7 (Tiger Lake) CPU. All algorithms have the same operational environment: Windows 11 on Intel Core i5-1135G7 (Tiger Lake). All algorithms are tested on the claimed operational environment.

The Jabber software calls the CiscoSSL FIPS Object Module (FOM) v7.3a that has been validated in accordance with the specified standards to meet the requirements listed below and all the algorithms claimed have CAVP certificates.

Platform/CPU	SFR	Algorithm	CAVP Certificate Number	Standard
Windows 11 on Intel Core i5-1135G7 (Tiger Lake)	FCS_CKM.1/KeyGen	RSA 2048 bits	A4446	FIPS Pub 186-4
Windows 11 on Intel Core i5-1135G7 (Tiger Lake)	FCS_CKM.1/KeyGen	ECC P-256/384/521	A4446	FIPS Pub 186-4
Windows 11 on Intel Core i5-1135G7 (Tiger Lake)	FCS_CKM.2	RSA 2048 bits	Tested with a known good implementation	NIST SP 800-56B
Windows 11 on Intel Core i5-1135G7 (Tiger Lake)	FCS_CKM.2	CVL-KAS-ECC P-256/384/521	A4446	NIST SP 800-56A
Windows 11 on Intel Core i5-1135G7 (Tiger Lake)	FCS_COP.1/SKC	CBC, GCM 128, 256 bits	A4446	FIPS Pub 197 NIST SP 800-38A NIST SP 800-38D
Windows 11 on Intel Core i5-1135G7 (Tiger Lake)	FCS_COP.1/SRTP	CTR, GCM 128, 256 bits	A4446	FIPS Pub 197 NIST SP 800-38A NIST SP 800-38D
Windows 11 on Intel Core i5-1135G7 (Tiger Lake)	FCS_COP.1/Hash	SHS SHA-1/256/384/512	A4446	FIPS Pub 180-4
Windows 11 on Intel Core i5-1135G7 (Tiger Lake)	FCS_COP.1/Sig	RSA 2048 bits	A4446	FIPS Pub 186-4
Windows 11 on Intel Core i5-1135G7 (Tiger Lake)	FCS_COP.1/KeyHash	HMAC SHA-256/384/512	A4446	FIPS Pub 198-1 FIPS Pub 180-4

TOE CAVP Certificates

2. Protection Profile SFR Assurance Activities

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

2.1 Communication (FCO)

2.1.1 Fixed-Rate Vocoder (VVoIPAS10:FCO_VOC_EXT.1)

2.1.1.1 VVoIPAS10:FCO_VOC_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 specifies each vocoder used. The evaluator shall then examine the specification for each vocoder in order to verify that no variable rate vocoders are claimed by the TSF.

Section 6.1 of the ST, FCS_VOC_EXT.1 states that the Cisco Jabber TOE transmits voice media using a constant bitrate (CBR) vocoder which ensures packet length is the same. The constant bitrate voice codecs implemented by the TSF are: G.711A, G.711u, G.722, G.722.1 and G.729.

The ITU-T standards indicate that all of these vocoders are fixed rate. The TOE does not claim any vocoders that are variable rate.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The evaluator shall set up a test environment that contains the TOE, a call control server (which may be the TOE itself), a network switch, a packet capture tool, and a second VVoIP endpoint.

The evaluator shall then perform the following test:

1. The evaluator shall ensure the TOE and the second VVoIP endpoint are registered to a call control server or are using P2P.

2. The evaluator shall use the TOE to dial the second VVoIP endpoint to establish a call and verify the call is established by holding a voice conversation.

3. The evaluator shall review the captured traffic to verify that a fixed rate vocoder is used.

If multiple vocoders are supported, the evaluator shall reconfigure the TOE to use each individual vocoder and repeat steps 1-3 for each vocoder.

The evaluator registered the TOE and a peer endpoint to a call control server, and confirmed the call control path. The vocoder is configured by the call control server; therefore evaluator configured the call control server to only offer one vocoder at a time by following the AGD, and placed a call between the endpoints while capturing the traffic. The evaluator tested the configuration of each vocoder specified in the TSS:

- G.711A
- G.711u
- G.722
- G.722.1
- G.729

The evaluator examined the resulting packet captures, and in each case, the evaluator determined that the configured fixed rate vocoder was used by the TOE.

2.2 Cryptographic support (FCS)

2.2.1 Cryptographic Asymmetric Key Generation - per TD0717 & TD0945 (ASPP14:FCS_CKM.1/AK)

2.2.1.1 ASPP14:FCS_CKM.1.1/AK

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

Section 6.1 of the ST, FCS_CKM.1/KeyGen states that to support TLS, the TOE implements a specified key generation algorithm to generate asymmetric cryptographic keys in accordance with RSA and ECC schemes that meet FIPS PUB 186-4. The key sizes are:

- RSA scheme: 2048-bit
- ECC using NIST curve of P-256, P-384, and P-521

The ST does not select 'invokes platform-provided functionality for asymmetric key generation'.

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

Section "CUCM Device - Softphone" of the AGD contains instructions to configure the key generation scheme and key size. This includes all selected schemes and key sizes. The selections made here control the key generation scheme and key sizes that the TOE will use for TLS communications by determining the parameters of the certificate it is issued for mutual authentication.

Component Testing Assurance Activities: If the application 'implements asymmetric key generation,' then the following test activities shall be carried out.

Evaluation Activity Note: The following tests may require the developer to provide access to a developer environment that provides the evaluator with tools that are typically available to end-users of the application.

Key Generation for FIPS PUB 186-4 or FIPS PUB 186-5 RSA Schemes

The evaluator shall verify the implementation of RSA Key Generation by the TOE using the Key Generation test. This test verifies the ability of the TSF to correctly produce values for the key components including the public

verification exponent e, the private prime factors p and q, the public modulus n and the calculation of the private signature exponent d. Key Pair generation specifies 5 ways (or methods) to generate the primes p and q. These include:

1. Random Primes:

Provable primes

Probable primes

2. Primes with Conditions:

Primes p1, p2, q1,q2, p and q shall all be provable primes

Primes p1, p2, q1, and q2 shall be provable primes and p and q shall be probable primes

Primes p1, p2, q1, q2, p and q shall all be probable primes

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

n = p*q,

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

GCD(p-1,e) = 1,

GCD(q-1,e) = 1,

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

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

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

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

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

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

Key Generation for Elliptic Curve Cryptography (ECC)

FIPS 186-4 or FIPS 186-5 ECC Key Generation Test For each supported NIST curve, i.e., P-256, P384 and P-521, the evaluator shall require the implementation under test (IUT) to generate 10 private/public key pairs. The private key shall be generated using an approved random bit generator (RBG). To determine correctness, the evaluator shall submit the generated key pairs to the public key verification (PKV) function of a known good implementation.

FIPS 186-4 or FIPS 186-5 Public Key Verification (PKV) Test For each supported NIST curve, i.e., P256, P-384 and P-521, the evaluator shall generate 10 private/public key pairs using the key generation function of a known good implementation and modify five of the public key values so that they are incorrect, leaving five values unchanged (i.e., correct). The evaluator shall obtain in response a set of 10 PASS/FAIL values. Key Generation for Finite-Field Cryptography (FFC) The evaluator shall verify the implementation of the Parameters Generation and the Key Generation for FFC by the TOE using the Parameter Generation and Key Generation test. This test verifies the ability of the TSF to correctly produce values for the field prime p, the cryptographic prime q (dividing p-1), the cryptographic group generator g, and the calculation of the private key x and public key y. The Parameter generation specifies 2 ways (or methods) to generate the cryptographic prime q and the field prime p:

Cryptographic and Field Primes:

Primes q and p shall both be provable primes

Primes q and field prime p shall both be probable primes

and two ways to generate the cryptographic group generator g:

Cryptographic Group Generator:

Generator g constructed through a verifiable process

Generator g constructed through an unverifiable process.

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

len(q) bit output of RBG where 1 =x = q-1

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

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

by comparing values generated by the TSF with those generated from a known good implementation. Verification must also confirm

g not= 0,1

q divides p-1

g^q mod p = 1

g^x mod p = y

for each FFC parameter set and key pair.

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

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

(TD0945 applied)

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

2.2.2 Cryptographic Key Establishment (ASPP14:FCS_CKM.2)

2.2.2.1 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, FCS_CKM.2 states that the TOE implements RSA-based key establishment schemes that meet 800-56B and ECC key establishment schemes that meet SP800-56A.

This aligns with the selections made in FCS_CKM.1/AK.

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 "CUCM Device - Softphone" of the AGD contains instructions to configure the key establishment schemes. The selections made here control the key establishment scheme that the TOE will use for TLS communications by determining the parameters of the certificate it is issued for mutual authentication.

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

RSA key establishment was tested using a known-good implementation, which was OpenSSL.

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

2.2.3.1 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.

Section 6.1 of the ST, FCS_CKM_EXT.1 states that the TOE implements asymmetric cryptography in support of TLS.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

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

2.2.4.1 ASPP14:FCS_COP.1.1/Hash

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.1 of the ST, FCS_COP.1/Hash states that the TOE provides cryptographic hashing services using SHA-1, SHA-256, SHA-384, and SHA-512 as specified in FIPS Pub 180-4 “Secure Hash Standard.”

SHA-256 and SHA-384 and SHA-512 are used in TLS to validate server certificates on establishing TLS connections. SHA-256 and SHA-384 and SHA-512 are used when the TOE generates asymmetric keys for Elliptic curve-based key establishment. SHA-1 is used with SRTP.

Component Guidance Assurance Activities: None Defined

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

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

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

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

Test 3: Selected Long Messages Test - Bit oriented Mode The evaluators devise an input set consisting of m messages, where m is the block length of the hash algorithm. The length of the ith message is 512 + 99*i, where 1

<= i <= m. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

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

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

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

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

2.2.5.1 ASPP14:FCS_COP.1.1/KeyedHash

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: 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.

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

2.2.6 Cryptographic Operation - Signing - per TD0717 & TD0945 (ASPP14:FCS_COP.1/Sig)

2.2.6.1 ASPP14:FCS_COP.1.1/Sig

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

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

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

ECDSA Algorithm Tests (TD945 applied)

Test 1: ECDSA FIPS 186-4 or FIPS 186-5 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 or FIPS 186-5 Signature Verification Test. For each supported NIST curve (i.e., P-256, P-384 and P-521) and SHA function pair, the evaluator shall generate a set of 10 1024-bit message, public key and signature tuples and modify one of the values (message, public key or signature) in five of the 10 tuples. The evaluator shall obtain in response a set of 10 PASS/FAIL values.

RSA Signature Algorithm Tests

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

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

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

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

2.2.7.1 ASPP14:FCS_COP.1.1/SKC

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

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

No configuration of the TOE is necessary to use the claimed cryptographic algorithms or key sizes. The TLS ciphersuites that utilize these encryption/decryption algorithms are enabled by default. Section "SIP Connections and Protocols" of the AGD states that there is no direct admin or user interaction on the TOE to configure or set the SRTP channel and no user configuration is required for the cryptographic features of the TOE.

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

AES-CBC Known Answer Tests

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

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

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

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

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

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

AES-CBC Multi-Block Message Test

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

AES-CBC Monte Carlo Tests

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

Input: PT, IV, Key

for i = 1 to 1000:

if i == 1:

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

PT = IV

else:

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

PT = CT[i-1]

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

AES-GCM Monte Carlo Tests

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

128 bit and 256 bit keys

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

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

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

The evaluator shall test the encrypt functionality using a set of 10 key, plaintext, AAD, and IV tuples for each combination of parameter lengths above and obtain the ciphertext value and tag that results from AES-GCM

authenticated encrypt. Each supported tag length shall be tested at least once per set of 10. The IV value may be supplied by the evaluator or the implementation being tested, as long as it is known.

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

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

AES-XTS Tests

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

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

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

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

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

AES-CCM Tests

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

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

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

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

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

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

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

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

Variable Associated Data Test

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

Variable Payload Test

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

Variable Nonce Test

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

Variable Tag Test

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

Decryption-Verification Process Test

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

AES-CTR Tests

Test 1: Known Answer Tests (KATs)

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

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

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

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

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

Test 2: Multi-Block Message Test

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

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

Test 3: Monte-Carlo Test

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

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

For AES-ECB mode

Input: PT, Key

for i = 1 to 1000:

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

PT = CT[i]

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

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

2.2.8 HTTPS Protocol (ASPP14:FCS_HTTPS_EXT.1/Client)

2.2.8.1 ASPP14:FCS_HTTPS_EXT.1.1/Client

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

Section 6.1 of the ST, FCS_HTTPS_EXT.1/Client states that the TOE implements HTTPS over TLS that complies with RFC2818. HTTPS is used to protect communications between the TOE and the CUCM server for authentication and configuration purposes. The HTTPS Client is implemented by the curl library which is bundled in the TOE. The curl library runs over the CiscoSSL FIPS Object Module (FOM) which is also bundled in the TOE. The curl library initiates a connection to the server on the appropriate port. When the TLS handshake has successfully finished, the curl

library initiates the first HTTP request. The X.509 certificate presented by the server endpoint is validated before establishing the secure connection. The TOE will not establish an HTTPS connection when the peer certificate is deemed to be invalid.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator shall attempt to establish an HTTPS connection with a webserver, observe the traffic with a packet analyzer, and verify that the connection succeeds and that the traffic is identified as TLS or HTTPS.

As part of testing for FIA_X509_EXT.1, the evaluator established a connection between the TOE HTTPS client and the CUCM webserver and captured the traffic. Through analysis of the packet capture, the evaluator was able to determine that the connection was successful and that the traffic was identified as TLS.

2.2.8.2 ASPP14:FCS_HTTPS_EXT.1.2/Client

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: Other tests are performed in conjunction with the TLS package.

As indicated by the Assurance Activity, other tests are performed in conjunction with FCS_TLSC_EXT.1 and/or FCS_TLSS_EXT.1. See those tests for details.

2.2.8.3 ASPP14:FCS_HTTPS_EXT.1.3/Client

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

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

Test 1: The evaluator shall demonstrate that using a certificate without a valid certification path results in the selected action in the SFR. If 'notify the user' is selected in the SFR, then the evaluator shall also determine that the user is notified of the certificate validation failure. Using the administrative guidance, the evaluator shall then load a certificate or certificates to the Trust Anchor Database needed to validate the certificate to be used in the function, and demonstrate that the function succeeds. The evaluator then shall delete one of the certificates, and

show that again, using a certificate without a valid certification path results in the selected action in the SFR, and if 'notify the user' was selected in the SFR, the user is notified of the validation failure.

'notify the user is not selected'.

This was tested alongside test cases FIA_X509_EXT.1.1 test 1 which cover the cases of valid and incomplete trust chains. The TLS connection was successful when the necessary certificate was trusted by the TOE, and unsuccessful when it was not.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

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

2.2.9.1 ASPP14:FCS_RBG_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

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

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

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

The 'invoke platform-provided DRBG functionality' selection is made in the requirement.

Section 6.1 of the ST, FCS_RBG_EXT.1 states that the TOE invokes the BCryptGenRandom API on the platform when needed to generate a cryptographic key.

This API is in the allowable list for the Windows platform in the ASPP.

The TSS states that this applies to the following SFRs:

FCS_CKM.1/KeyGen - Cryptographic Asymmetric Key Generation

FCS_CKM.2 - Cryptographic Key Establishment

FCS_SRTP_EXT.1 - Secure Real-Time Transport Protocol

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

2.2.10 Secure Real-Time Transport Protocol - per TD0965 (VVoIPAS10:FCS_SRTP_EXT.1)

2.2.10.1 VVoIPAS10:FCS_SRTP_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.10.2 VVoIPAS10:FCS_SRTP_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.10.3 VVoIPAS10:FCS_SRTP_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.10.4 VVoIPAS10:FCS_SRTP_EXT.1.4

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 verify that it describes how the SRTP session is negotiated for both incoming and outgoing calls. This includes how the keying material is established, as well has how requests to use the NULL algorithm or other unallowed cipher suites are rejected by the TSF.

Section 6.1 of the ST, FCS_SRTP_EXT.1 states that the TOE implements secure RTP (SRTP) as defined in RFC 3711 to provide encryption and authentication of audio and video data streams between itself and a VVoIP endpoint. The TOE uses AES encryption/decryption supporting the following ciphersuites:

- AEAD_AES_256_GCM

For incoming and outgoing calls, SRTP keys are exchanged as defined in section 4 of RFC 4568. The TOE uses the Session Description Protocol (SDP) within SIP signaling messages as the “inline” parameter within SDP packets. SRTP key exchange is accomplished through the "crypto" attribute (a=crypto) under the particular media type (m=video or m=audio) within the SDP. The format of the "crypto" attribute is shown below.

a=crypto:<tag> <crypto-suite> <key-params> [<session-params>]

The tag field is a decimal number used to uniquely distinguish multiple crypto attributes from each other. The crypto-suite field contains the cipher and message authentication algorithms for the particular crypto attribute. The key-params field itself has the format shown below.

key-params = <key-method> ":" <key-info>

The key-method field indicates the method by which keying material is exchanged and 'inline' indicates the keying information is contained within the SDP. An example of the use of SDP for SRTP is shown below:

m=video 16386 RTP/SAVP 112

c=IN IP4 14.50.248.31

a=crypto:1 AEAD_AES_256_GCM

inline:8V0Oj7duRGA1lxd1At5eYSpc8Q1qIR4o0Qq3Sr74ZR5+j3eFmro4CvPDvQQ=

Since keying material is sent within SIP signaling, encryption of the SIP signaling is required to secure the key exchange. The TOE implements TLS to protect the SRTP key exchange between itself and the CUCM server. The CUCM SIP Server is responsible for the configuration of the policy regarding which ciphers are allowed. The TOE cannot override this configuration. The evaluated configuration requires the administrator of the CUCM SIP Server to configure only 'secure/encrypted' calls. By doing so attempts to call using SRTP NULL encryption fail.

Component Guidance Assurance Activities: The evaluator shall examine the operational guidance to determine that it includes instructions for how to disable the use of the SRTP NULL algorithm and how to specify the ports to be used for SRTP communications.

Section "SIP Connections and Protocols" in the AGD states that there is no direct admin or user interaction on the TOE to configure or set the SRTP channel and no user configuration is required for the cryptographic features of the TOE. The CUCM SIP Server administrator configures the required settings appropriately and then each time a call is made the TOE automatically starts SRTP streams are negotiated. To ensure the SRTP AEAD_AES_256_GCM

cipher is used, the CUCM Administrator must configure SRTP ciphers to select the “Strongest Cipher” setting only. There is no user or admin interaction per-SRTP-channel.

Component Testing Assurance Activities: The evaluator shall follow the procedure for initializing their device so that they are ready to receive and place calls. For each cipher suite selected in FCS_SRTP_EXT.1.2, the evaluator shall configure the SIP server to only allow that cipher suite to be used. The evaluator shall then both place and receive a call and determine that the traffic sent and received by the TOE is encrypted using SRTP with that cipher suite. The evaluator may choose one of the below two options to ensure that the call is being encrypted and to view the cipher suite being used.

Option 1: The evaluator shall configure the SIP server to report whether SRTP is being used, and if so, print the negotiated SRTP cipher suite. The evaluator shall confirm that SRTP was used for the calls and that the correct cipher suite was negotiated.

Option 2: A packet capture tool should be used with the SIP server's private key loaded in. The evaluator shall decrypt the TLS-SIP traffic, view the SDES negotiation, and ensure that the correct cipher suite was negotiated.

Option 3: The evaluator shall inspect the VVOIP endpoint's audit or system log to ensure that the correct cipher suite was negotiated.

Next, the evaluator shall configure the SIP server to only allow the SRTP NULL cipher suite. The evaluator shall attempt to both place and receive a call and confirm that both attempts failed.

(TD0965 applied)

The evaluator configured the TOE such that it was ready to place and receive calls. The TOE supports AEAD_AES_256_GCM, therefore that is the algorithm that was tested. The SIP server was configured to only allow this algorithm by following the AGD. The evaluator additionally configured the SIP server to audit the negotiated SRTP ciphersuite (i.e., Option 1).

The evaluator placed a call from the TOE, held a brief conversation, then ended the call. The evaluator followed up by receiving a call on the TOE (by placing the call from a peer endpoint) and performed the same actions. After the calls were completed, the evaluator collected and examined logs the from the SIP Server.

In both cases of placing and receiving a call from the TOE, the SIP server logged information which indicated that it negotiated the correct SRTP cipher, only allowing the configured AEAD_AES_256_GCM ciphersuite.

For the SRTP NULL ciphersuite test, the evaluator configured the TOE to communicate with a SIP server that only allowed the SRTP NULL ciphersuite. With this configuration, the evaluator attempted to place a call to and from the TOE and found that the call did not establish on either endpoint.

2.2.11 Storage of Credentials - per TD0865 (ASPP14:FCS_STO_EXT.1)

2.2.11.1 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, FCS_STO_EXT.1 states that the Cisco Jabber TOE leverages the platform for securely storing persistent credentials. The TOE uses the Windows Data Protection API (DPAPI) to store User password (secret). The TOE uses Windows Certificate Manager to store the private key used for X.509 certificate generation and peer authentication.

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 the case of certificates, the evaluator examined the Windows Certificate Store on the TOE platform and found that the certificate issued to the TOE by the ESC was stored there. For the case of credentials such as passwords which are used by the TOE, the evaluator used the Microsoft Windows tool WinDbg to determine that the DPAPI was being invoked to store credentials. This indicates that the TOE is correctly using the claimed methods to store sensitive data.

2.2.12 TLS Protocol (PKGTLS11:FCS_TLS_EXT.1)

2.2.12.1 PKGTLS11:FCS_TLS_EXT.1.1

TSS Assurance Activities: None Defined

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

The ST claims that the TOE is a TLS Client. This is consistent with the presence of PKGTLS11:FCS_TLSC_EXT.* SFRs and their corresponding selections in the ST.

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.13 TLS Client Protocol (PKGTLS11:FCS_TLSC_EXT.1)

2.2.13.1 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, FCS_TLSC_EXT.1 states that the Cisco Jabber TOE implements TLS 1.2 with the following ciphersuites:

TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384

TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384

TLS connections related to CAPF exclusively use TLS_RSA_WITH_AES_128_CBC_SHA. The other evaluated TLS connections make use of the other two cipher suites.

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 "SIP Connections and Protocols" of the AGD states that there is no direct admin or user interaction on the TOE to configure or set the SRTP channel and no user configuration is required for the cryptographic features of the TOE.

The TOE does not need explicit configuration to meet the requirement.

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.

Test 1: The evaluator attempted to establish a TLS session using each of the cipher suites specified by the requirement. The evaluator observed that the TOE accepted the connections using the cipher suites claimed in the Security Target.

Test 2: The evaluator connected the TOE to a test server which uses a valid certificate in the TLS connection. The TOE successfully connected. The evaluator then connected the TOE to a test server which uses a certificate missing the Server Authentication key usage. The TOE rejected the connection.

Test 3: The evaluator attempted to connect the TOE to a test server that returns a server certificate in the TLS connection that does not match the server-selected cipher suite. The TOE rejected the connection.

Test 4: The evaluator connected the TOE to a test server that attempted to use the NULL ciphersuite with TLS. The TOE rejected the connection.

Test 5.1: The evaluator connected the TOE to a test server that selected an undefined TLS version during the handshake. The TOE rejected the connection.

Test 5.2: The evaluator connected the TOE to a test server that reported an unsupported recent TLS version. The TOE rejected the connection.

Test 5.3: The evaluator connected the TOE to a test server with a TLS connection in which the server nonce was modified. The TOE rejected the connection.

Test 5.4: The evaluator connected the TOE to a test server with a TLS connection in which the ciphersuite is not one matching a ciphersuite in the Client Hello message. The TOE rejected the connection.

Test 5.5: The evaluator connected the TOE to a test server with a TLS connection in which the server's key exchange signature block was modified. The TOE rejected the connection.

Test 5.6: The evaluator connected the TOE to a test server with a TLS connection in which the server's Server Finished message was modified. The TOE rejected the connection.

Test 5.7: The evaluator connected the TOE to a test server with a TLS connection in which the server sent a garbled message after the Change Cipher Spec message. The TOE rejected the connection and no application data was present.

2.2.13.2 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, FCS_TLSC_EXT.2 states that the TOE compares the FQDN of the server it is establishing connectivity with against the Subject Alternate Name-dnsName attributes in the certificate. If Jabber determines there is a mismatch, it will not establish the TLS trusted channel. This applies to all TLS connections. The TOE does not support the use of wildcards within certificates and does not support certificate pinning.

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 SIP Connection and Protocols in the AGD states that for setting the reference identifier to establish TLS connection to CUCM refer to the “Service Discovery Options” and “Manual Connection Settings” sections in the “Configure Service Discovery” chapter in On-Premise Deployment for Cisco Jabber 14.1 [2]. Cisco Jabber compares the FQDN of the CUCM server against the Subject Alternate Name-dnsName attributes in the certificate. If Jabber determines there is a mismatch, it will not establish the TLS trusted channel.

Note: If the server certificate does not contain a SAN, the CN of the certificate is used for comparison purposes instead.

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.

Test 1: The evaluator attempted to connect the TOE to a test server in which the test server's certificate contains a CN that does not match the reference identifier expected by the TOE and does not contain the SAN extension. The TOE rejected the connection.

Test 2: The evaluator attempted to connect the TOE to a test server in which the test server's certificate contains a CN that matches the reference identifier, but also contains a SAN that does not match the reference identifier expected by the TOE. The TOE rejected the connection.

Test 3: The evaluator attempted to connect the TOE to a test server in which the test server's certificate contains a CN that matches the reference identifier expected by the TOE, but is missing a SAN. The TOE accepted the connection.

Test 4: The evaluator attempted to connect the TOE to a test server in which the test server's certificate contains a CN that does not match the reference identifier but does contain an identifier in the SAN that matches. The TOE accepts the connection.

Test 5 (5.1 through 5.3): Not applicable, as the TOE does not support wildcards.

Test 5.4: The evaluator attempted to connect the TOE to a test server in which the test server's certificate contains a reference identifier with a wildcard in the left-most label (for example, connecting the TOE to test.example.com, with a certificate that indicates *.example.com). The TOE rejected the connection.

Test 6: This test is not applicable as the TOE does not support URI or Service name reference identifiers.

Test 7: This test is not applicable as the TOE does not support pinned certificates.

2.2.13.3 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.

The selection for override of invalid certificates is not made. This TSS Assurance Activity is not applicable.

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 1a and 1b: The valid and invalid certificate chain tests are performed in PKGTLS11:FIA_X509_EXT.1. The TOE accepts a certificate with a valid path, and rejects a certificate if the path is incomplete.

Test 1c: This test is not applicable as the TOE trust store is provided and managed by the platform.

Test 2: The revoked certificate tests are performed in PKGTLS11:FIA_X509_EXT.1. The TOE rejects certificates that are revoked.

Test 3: The expired certificate tests are performed in PKGTLS11:FIA_X509_EXT.1. The TOE rejects certificates that are expired.

Test 4: The reference identifier tests are performed in PKGTLS11:FCS_TLSC_EXT.1.2. The TOE rejects certificates that do not contain a valid identifier.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.14 TLS Client Support for Mutual Authentication (PKGTLS11:FCS_TLSC_EXT.2)

2.2.14.1 PKGTLS11:FCS_TLSC_EXT.2.1

TSS Assurance Activities: The evaluator shall ensure that the TSS description required per FIA_X509_EXT.2.1 includes the use of client-side certificates for TLS mutual authentication. The evaluator shall also ensure that the TSS describes any factors beyond configuration that are necessary in order for the client to engage in mutual authentication using X.509v3 certificates.

Section 6.1 of the TSS, FCS_TLSC_EXT.2 states that to obtain a X.509 certificate, the Cisco Jabber TOE must securely enroll in the Certificate Authority Proxy Function (CAPF) as instructed in the AGD. CAPF runs on the Cisco Unified Communications Manager and provides the TOE with X.509 certificates. The CAPF process also determines the certificate the TOE will use to establish a TLS connection. TLS mutual authentication is supported for the SIP connection between the TOE and CUCM Server.

Guidance Assurance Activities: The evaluator shall ensure that the AGD guidance includes any instructions necessary to configure the TOE to perform mutual authentication. The evaluator also shall verify that the AGD guidance required per FIA_X509_EXT.2.1 includes instructions for configuring the client-side certificates for TLS mutual authentication.

Section "SIP Connections and Protocols" in the AGD states that by default, the TOE supports TLS mutual authentication - no configuration is required to enable this feature. TLS mutual authentication ensures that only Client Services Framework (CSF) devices (i.e. the TOE) with the correct certificates can register to CUCM SIP Server. Likewise, CSF devices can register only to CUCM SIP Server instances that provide the correct certificate.

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

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

Test 2: The evaluator shall establish a connection to a server with a shared trusted root that is configured for mutual authentication (i.e. it sends Server's Certificate Request (type 13) message). The evaluator observes

negotiation of a TLS channel and confirms that the TOE responds with a non-empty Client's Certificate message (type 11) and Certificate Verify (type 15) message.

This test was performed on the SIP channel. Other channels were excluded because the SIP channel is the only one that supports mutual authentication.

Test 1: The evaluator configured the test server to not require mutual authentication and attempted to connect the TOE to it. The evaluator verified that the TOE did not send the Client's Certificate message (type 11) during the handshake.

Test 2: The evaluator configured the test server to require mutual authentication and attempted to connect the TOE to it. The evaluator verified that the TOE responded with a non-empty Client's Certificate message (type 11) and Certificate Verify (type 15) message.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

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

2.2.15.1 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.

Test 1: The evaluator took a packet capture of a successful connection between the TOE and a TLS server. The evaluator searched for both the SCSV cipher suite and the renegotiation info field. The evaluator found that the TOE supports the SCSV ciphersuite in the Client Hello.

Test 2: The result of Test 1 shows that the Client successfully handles ServerHello messages received during the initial handshake that include the 'renegotiation_info' extension, as the Client successfully establishes a connection after checking that the ServerHello message's renegotiation_info is correctly formed. The evaluator then used a TLS server that modified the length portion of the ServerHello message to a non-zero value. The TOE correctly rejected the packet and closed the connection.

Test 3: The evaluator used a TLS server that corrupted data in the client_verify_data field. The TOE recognized the invalid value and correctly rejected the connection attempt.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.16 TLS Client Support for Supported Groups Extension (PKGTLS11:FCS_TLSC_EXT.5)

2.2.16.1 PKGTLS11:FCS_TLSC_EXT.5.1

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

Section 6.1 of the TSS, FCS_TLSC_EXT.5 states that the TOE implements secp256r1, secp384r1, secp521r1 elliptic curve extensions in the supported_groups extension of the Client Hello packet. This is the behavior by default and is not configurable.

Guidance Assurance Activities: None Defined

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

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

This test was performed on the HTTPS and SIP channels which support secp groups. The CAPF channel was excluded as it only utilizes RSA key exchange.

Test 1: The evaluator connected the TOE to the ESC using each of the supported curves. The TOE connects successfully using each claimed curve.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.3 User data protection (FDP)

2.3.1 Encryption Of Sensitive Application Data - per TD0756 (ASPP14:FDP_DAR_EXT.1)

2.3.1.1 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 6.1 of the ST, FDP_DAR_EXT.1 states that the TOE utilizes the following sensitive data:

- The private key used for X.509 certificate generation and peer authentication
- User password (secret)

Each is protected in accordance with FCS_STO.EXT.1

The Cisco Jabber TOE leverages the platform for securely storing persistent credentials. Specifically, the TOE uses the Windows Data Protection API (DPAPI) to store User password (secret), and TOE uses Windows Certificate Manager to store the private key used for X.509 certificate generation and peer authentication.

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.

The Security Target selects "protect sensitive data in accordance with FCS_STO_EXT.1". Therefore, all sensitive data used by the TOE is covered in FCS_STO_EXT.1.

See testing for FCS_STO_EXT.1 which demonstrates that the Windows Certificate Store is used to store keys, and that credentials are encrypted and stored using Windows DPAPI.

2.3.2 Access to Platform Resources - per TD0822 (ASPP14:FDP_DEC_EXT.1)

2.3.2.1 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 6.1 of the ST, section FDP_DEC_EXT.1 states that the Cisco Jabber TOE restricts access to the following network connectivity resources: camera, micro-phone, USB, and Bluetooth. To function as a VVoIP product, access to network connectivity, the camera, and microphone resources are required. In addition, the TOE can access Bluetooth or USB hardware resources to control Bluetooth or USB audio devices.

This description is consistent with the selections made in the requirement and provides sufficient justification.

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 AppxManifest.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.

(TD0822 applied)

The Security Target states that "The Cisco Jabber TOE restricts access to the following network connectivity resources: camera, microphone, USB, and Bluetooth. To function as a VVoIP product, access to network connectivity, the camera, and microphone resources are required."

Therefore, the TOE documentation (ST) lists the required hardware resources.

2.3.2.2 ASPP14:FDP_DEC_EXT.1.2

TSS Assurance Activities: None Defined

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

Section 6.1 of the ST, section FDP_DEC_EXT.1 states that the TOE restricts its access to the following repositories:

- Address book and calendar - Needed to import contacts and for calling VVoIP users.
- File-system - To import contacts, access to the file system is needed.
- Photo Library - Users may import a photo from the photo library to be used on VVoIP calls.

This description is consistent with the selections made in the requirement and provides sufficient justification for each sensitive information repository.

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 AppxManifest.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.

(TD0822 applied)

The Security Target states the following:

The TOE restricts its access to the following repositories:

- Address book and calendar - Needed to import contacts and for calling VVoIP users.
- File-system - To import contacts, access to the file system is needed.
- Photo Library - Users may import a photo from the photo library to be used on VVoIP calls.

Therefore, the TOE documentation (ST) lists the sensitive information repositories that it accesses. Access to each of these must be initiated by the user of the TOE.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.3.3 Subset Information Flow Control (VVoIPAS10:FDP_IFC.1)

2.3.3.1 VVoIPAS10:FDP_IFC.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 describes how streaming media is not transmitted when not in a streaming media state.

Section 6.1 of the TSS, FDP_IFC.1 states that the TOE enforces a media transmission policy that ensures media data is not transmitted unless it is in a streaming media state as defined in the rules for FDP_IFF.1.

The referenced rules for being in the streaming media state as described by the TSS are (all must be met):

- The TOE is registered with the ESC
- A call has been established with a telephony device (VVoIP endpoint),
- The TOE is in the off-hook state,
- The TOE is not in the mute state,
- The TOE is not in the hold state

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: Testing of this component is performed through evaluation of FDP_IFF.1.

Testing of this component is performed through evaluation of FDP_IFF.1.

2.3.4 Simple Security Attributes - per TD0938 (VVoIPAS10:FDP_IFF.1)

2.3.4.1 VVoIPAS10:FDP_IFF.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.4.2 VVoIPAS10:FDP_IFF.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.4.3 VVoIPAS10:FDP_IFF.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.4.4 VVoIPAS10:FDP_IFF.1.4

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.4.5 VVoIPAS10:FDP_IFF.1.5

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 describes the TOE's enforcement of the media transmission policy and describes the conditions that are necessary for the TSF to transmit voice/video data to the operational environment.

See the previous TSS entry for FDP_IFF.1 which also addresses the TOE's media transmission policy and the conditions for the TSF transmitting voice/video data.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The evaluator shall set up a test environment that contains the TOE, a call control server (which may be the TOE itself), a network switch, a traffic capture tool, and a second VVoIP endpoint.

The evaluator shall then perform the following tests:

Test 1-App (conditional: software application TOE):

1. The evaluator shall execute the VVoIP application without registering it to the call control server or using P2P. The evaluator shall use the traffic capture tool to verify that the TOE does not transmit any streaming media traffic.

2. The evaluator shall place the TOE into the off-hook state (e.g., by performing a call without specifying any destination data). The evaluator shall use the traffic capture tool to verify that the TOE does not transmit any streaming media traffic.

Test 2:

1. The evaluator shall ensure the TOE is using P2P or register the TOE with the call control server and verify that the TOE is registered by checking the call control server screen with current TOE connections and by viewing the call control path traffic using the traffic capture tool.

2. The evaluator shall place the TOE into the on-hook state and verify using the traffic capture tool that no streaming media traffic is transmitted by the TOE.

3. The evaluator shall place the TOE into the off-hook state. The evaluator shall then verify that the TOE continues to not transmit streaming media traffic.

Test 3:

1. The evaluator shall ensure the TOE is using P2P or shall register the TOE with the call control server and verify that it is registered by checking the call control server with current connections and using the traffic capture tool to verify the call control path traffic.

2. The evaluator shall ensure the TOE is using P2P or shall register the second VVoIP endpoint with the call control server and verify that it has been registered by checking the call control server with current connections and using the traffic capture tool to verify the call control path traffic.

3. The evaluator shall use the TOE to dial the second VVoIP endpoint and connect a call. The evaluator shall verify that the connection is made by having a voice/video conversation with the endpoint and using the traffic capture tool to verify that a steady stream of traffic is being transmitted between the two endpoints over the media channel.

4. The evaluator shall use the TOE to put the call on mute and verify that no traffic is transmitted from the TOE over the media channel to the second VVoIP endpoint or only silence packets are observed on the media channel. If the TOE is registered to a call control server, the evaluator shall also verify that a mute control message is sent to the call control server and it responds. (TD0938 applied)

5. The evaluator shall use the TOE to take the call off mute and verify that the streaming media traffic between the TOE and the second VVoIP endpoint is resumed.

Test 4:

1. The evaluator shall ensure the TOE is using P2P or register the TOE to the call control server and place it in the on-hook state.

2. The evaluator shall use a fuzzing tool to attempt to connect to the TOE on the full range of TCP ports used by the TSF. All ports used by the TOE should be closed except for the port that is used to communicate with the ESC.

Test 5:

1. The evaluator shall ensure the TOE and the second VVoIP endpoint are registered to a call control server or are using P2P.

2. The evaluator shall place the TOE in the on-hook state.

3. The evaluator shall use a fuzzing tool to attempt to connect to the TOE on the full range of UDP ports used by the TSF. All ports used by the TOE should be closed.

4. The evaluator shall place a call to the second VVoIP endpoint and verify the call is established. The evaluator shall capture the traffic to determine the port used by the TOE to carry the media traffic.

5. The evaluator shall hang up the call and verify that the TOE has returned to the on-hook state.

6. The evaluator shall perform fuzzing activities to verify that the port used to carry media traffic in step 4 has been closed.

Test 6 (conditional: 'The TOE is not in the hold state' is selected in FDP_IFF.1.2):

1. The evaluator shall use the TOE to put the call on hold and verify that no streaming media traffic is transmitted from the TOE over the media channel. If the TOE is registered to a call control server, the evaluator shall also verify that the VVoIP endpoint on-hold call control is sent to the call control server and it responds.

2. The evaluator shall use the TOE to take the call off hold and verify that the streaming media traffic between the TOE and the second VVoIP endpoint is resumed.

Test 1: After installing the TOE, the evaluator executed the Jabber Windows application. The evaluator left the TOE in this default state where it was unregistered to the call control server. The evaluator then began a packet capture, capturing traffic from the TOE. The evaluator then verified that basic network connectivity between the TOE and the call control server was functioning, but still did not register the TOE to the call control server. The evaluator then waited, leaving the TOE in its default pre-registration state to determine if it would send any streaming media traffic. After enough time had passed, the evaluator examined the packet capture. No traffic was observed from the TOE, including the lack of any streaming media traffic.

Prior to registration, the evaluator looked for any method of placing the TOE in the off-hook state and found none. Note that test 2 below which demonstrates placing the TOE in the off-hook state after registration has completed. Since there is no method of placing the TOE on the off-hook state prior to registration, there is therefore no mechanism to send streaming media traffic before registration.

Test 2: The evaluator registered the TOE with the call control server. After this process was complete, the TOE application indicated that the registration was complete. The evaluator followed up by checking the call control server and verified that registration was indicated as complete on its side. The evaluator then placed a call to a peer endpoint and confirmed that the TOE was using the correct call control path.

The evaluator placed the TOE in the "on-hook" state. The evaluator then waited, leaving the TOE in its "on-hook" state to determine if it would send any streaming media traffic. After enough time had passed, the evaluator examined the packet capture. The only traffic identified from the TOE during this time was TLS-protected SIP traffic between the TOE and the call control server. No streaming media traffic was observed while the TOE was in the "on-hook" state.

The evaluator placed the TOE in the "off-hook" state. The evaluator waited here, leaving the TOE in its "off-hook" state to determine if it would send any streaming media traffic. Similar to the previous "on-hook" case, no streaming media traffic was observed. The only difference in this case was that more TLS-protected SIP traffic was observed because of the active call. The same passing determination described previously also applies here.

Test 3: The evaluator once again confirmed the TOE registration to the call control server and verified that it was using the correct call control path.

The evaluator then placed a video call from the TOE endpoint to the peer endpoint. The evaluator observed that the call connected successfully and that the webcam was on the TOE endpoint on to cause streaming media traffic. The evaluator captured the traffic between the two endpoints and observed that a steady stream of streaming media traffic was sent between the two endpoints while the call was active.

The evaluator placed another call following the same procedure (except this time using the microphone instead of the camera as required by the test). The call began with the TOE endpoint unmuted, and the peer endpoint muted. The evaluator then briefly muted, the TOE, and then unmuted it. The call was then ended. The evaluator captured the traffic between the two endpoints and observed that the TOE correctly sent control messages to the call control server corresponding with the mute/unmute actions. Additionally, the TOE correctly stopped sending voice data (streaming media traffic) while it was muted, and the TOE continued to send it while the TOE was unmuted.

Test 4: The evaluator once again verified that the TOE was registered and using the correct call control path. The evaluator then performed an NMAP scan of the Windows machine hosting the TOE to determine the open ports. The scan reported a number of open TCP ports on the Windows system. Upon further examination, the evaluator determined that these ports were opened by the Windows platform by reviewing Microsoft's documentation. These ports were not opened by the TOE VVOIP application.

Test 5: The evaluator once again verified that the TOE was registered and using the correct call control path. The evaluator placed the TOE in the "on-hook" state, then performed an NMAP scan of the open UDP ports on the Windows platform hosting the TOE. The results were similar as the previous test case, in which the evaluator observed that a number of ports were open, but determined that these were opened by the Windows platform and not the TOE VVOIP application.

The evaluator then placed a call from the TOE to the peer endpoint, capturing the traffic between the two endpoints. The evaluator determined that the call was successful. Additionally, the UDP port in use during the call by the TOE was recorded. It was also observed that this dynamic port fell within the allowable range of UDP ports specified by the TSS for use in streaming media traffic. The evaluator then ended the call and determined that the TOE was back in the "on-hook" state again.

Following this, the evaluator performed the same scan as before, expecting that UDP port that was opened during the call to no longer be open. The evaluator observed that no ports other than the previously open ones were identified by the NMAP scan.

Test 6: This conditional test is included because the corresponding Security Target selection is made. The evaluator connected a call between the two endpoints. After allowing the call to establish, the evaluator then started a packet capture. The evaluator then placed the call on hold. The evaluator observed that the hold action was successful, and that the TOE sent a corresponding encrypted SIP message to the call control server, which responded.

With the call on hold, the evaluator resumed the packet capture between both endpoints. The call was then taken off hold. The evaluator examined the resulting packet capture and determined that before the hold was ended, no streaming media data was sent between the endpoints. Once the hold was ended, the streaming media traffic in UDP between the two endpoints resumed.

2.3.5 Network Communications (ASPP14:FDP_NET_EXT.1)

2.3.5.1 ASPP14:FDP_NET_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

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

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

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

Platforms: Android....

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

Test 1: The evaluator ran the TOE application and captured all traffic associated with the TOE. The evaluator examined this traffic and compared it to the traffic that is documented in the TSS for FDP_NET_EXT.1. The evaluator found that all of the network communications used by the TOE were covered in this TSS section.

Test 2: The evaluator examined the TCP and UDP ports that were open on the TOE platform. The evaluator determined that any ports open on the system were opened by the Windows platform and not the TOE itself. Additionally, any ports used by the TOE for communication to the call control server, or a peer endpoint were closed when not actively in use.

2.4 Identification and Authentication (FIA)

2.4.1 X.509 Certificate Validation - per TD0780 (ASPP14:FIA_X509_EXT.1)

2.4.1.1 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 6.1 of the ST, FIA_X509_EXT.1 states that the Cisco Jabber TOE invokes platform functionality to perform the following certificate validation:

- Certificate validation and certificate path validation according to RFC 5280.
- The certificate path must terminate with a trusted CA certificate.
- All CA certificates must have the basicConstraints extension present and be of type CA=TRUE.
- The certificate must not be revoked. Revocation is validated via the client device platform by OCSP and CRL revocation status check.
- The certificate must not be expired.
- The extendedKeyUsage field must be valid based on the following rules:

o Certificates used for trusted updates and executable code integrity verification shall have the Code Signing purpose (id-kp 3 with OID 1.3.6.1.5.5.7.3.3) in the extendedKeyUsage field.

o Server certificates presented for TLS shall have the Server Authentication purpose (id-kp 1 with OID 1.3.6.1.5.5.7.3.1) in the extendedKeyUsage field.

o Client certificates presented for TLS shall have the Client Authentication purpose (id-kp 2 with OID 1.3.6.1.5.5.7.3.2) in the extendedKeyUsage field.

o S/MIME certificates presented for email encryption and signature shall have the Email Pro-tection purpose (id-kp 4 with OID 1.3.6.1.5.5.7.3.4) in the extendedKeyUsage field.

o OCSP certificates presented for OCSP responses shall have the OCSP Signing purpose (id-kp 9 with OID 1.3.6.1.5.5.7.3.9) in the extendedKeyUsage field.

o Server certificates presented for EST shall have the CMC Registration Authority (RA) purpose (id-kp-cmcRA with OID 1.3.6.1.5.5.7.3.28) in the extendedKeyUsage field

The TOE relies upon the client device platform to verify the certificate at the point in time when it receives the server certificate as part of the process of establishing a secure connection to any server.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The tests described must be performed in conjunction with the other certificate services evaluation activities, including the functions in FIA_X509_EXT.2.1. The tests for the extendedKeyUsage rules are performed in conjunction with the uses that require those rules. If the application supports chains of

length four or greater, the evaluator shall create a chain of at least four certificates: the node certificate to be tested, two Intermediate CAs, and the self-signed Root CA. If the application supports a maximum trust depth of two, then a chain with no Intermediate CA should instead be created.

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

- by establishing a certificate path in which one of the issuing certificates is not a CA certificate,

- by omitting the basicConstraints field in one of the issuing certificates,

- by setting the basicConstraints field in an issuing certificate to have CA=False,

- by omitting the CA signing bit of the key usage field in an issuing certificate, and

- by setting the path length field of a valid CA field to a value strictly less than the certificate path.

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

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

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

The evaluator shall test revocation of the node certificate.

The evaluator shall also test revocation of an intermediate CA certificate (i.e. the intermediate CA certificate should be revoked by the root CA), if intermediate CA certificates are supported. If OCSP stapling per RFC6066 is the only supported revocation method, this test is omitted.

The evaluator shall ensure that a valid certificate is used, and that the validation function succeeds. The evaluator then attempts the test with a certificate that has been revoked (for each method chosen in the selection) to ensure when the certificate is no longer valid that the validation function fails.

Test 4: If any OCSP option is selected, the evaluator shall configure the 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.

Test 1: The evaluator connected the TOE to a test server using TLS and configured to present the following invalid certificates. With each certificate, the TOE TLS client successfully identified the certificate as invalid and refused the connection attempt.

- a certificate path in which one of the issuing certificates is not a CA certificate

- missing basicConstraints field in one of the issuing certificates

- basicConstraints field in an issuing certificate to have CA=False

- CA signing bit of the key usage field in an issuing certificate is missing

- a certificate with a path length field of a valid CA field set to a value strictly less than the certificate path

Note that the "certificate path in which one of the issuing certificates is not a CA certificate" test is covered by the missing basicConstraints and CA=false test.

Test 2: The evaluator configured a test server to use TLS. The test server was configured to present expired certificates. When the TOE TLS client attempted to connect, the connection was refused by the TOE.

Test 3: The evaluator configured a test sever to use TLS. The test server was configured with valid certificates allowing the TOE devices to connect to the test server. A successful connection was made by the TOE TLS client. The test server was then configured with a CRL revoked certificate. When the TOE attempted to connect, it rejected the connection. The test server was also configured with an OCSP revoked certificate, and the TOE similarly rejected the connection.

Test 4: The evaluator configured a test server to use TLS. The test server was configured with a certificate chain such that the server certificate CRL response was signed by a certificate that lacks the CRLSign purpose. The TOE correctly detects the invalid CRL signer's certificate and immediately rejected the connection attempt. The evaluator repeated this process with OCSP, with a response signed by a certificate that did not have the OCSP signing purpose. Similarly, the TOE rejected this connection attempt.

Test 5: The evaluator configured a test server to use TLS. The test server sends a modified certificate with a corrupted ASN.1 header. The TOE TLS client recognized that the certificate was invalid and rejected the connection attempt.

Test 6: The evaluator configured a test server to use TLS. The test server sends a modified certificate with a corrupted signature. The TOE TLS client recognized that the certificate was invalid and rejected the connection attempt.

Test 7: The evaluator configured a test server to use TLS. The test server sends a modified certificate with a corrupted public key. The TOE TLS client recognized that the certificate was invalid and rejected the connection attempt.

Tests 8 and 9: These conditional tests are not applicable, as the Security Target does not indicate support for EC certificates in FCS_COP.1/Sig.

2.4.1.2 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.

Test 1: This has been tested in ASPP14:FIA_X509_EXT.1.1, where the test server was configured with a certificate that did not contain the basicConstraints extension. When attempting to connect, the TOE rejected the connection when presented with this certificate.

Test 2: This has been tested in ASPP14:FIA_X509_EXT.1.1, where the test server was configured with a certificate in which the cA flag in the basicConstraints extension is set to FALSE. When attempting to connect, the TOE rejected the connection when presented with this certificate.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.4.2 X.509 Certificate Authentication (ASPP14:FIA_X509_EXT.2)

2.4.2.1 ASPP14:FIA_X509_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.4.2.2 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 6.1 of the ST, FIA_X509_EXT.2 states that to obtain a X.509 certificate, the Cisco Jabber TOE must securely enroll in the Certificate Authority Proxy Function (CAPF) as instructed in the AGD. CAPF runs on the Cisco Unified Communications Manager and provides the TOE with X.509 certificates. The CAPF process also determines the certificate the TOE will use to establish a TLS connection. TLS mutual authentication is supported for the SIP connection between the TOE and CUCM Server.

This section further states that if the TOE platform is unable to connect to the revocation server for example due to a network error, the TOE will not allow the certificate to be accepted.

This section does not call out any distinct differences in behavior of any of the trusted channels, indicating that all of them behave in the same manner.

The requirement does not indicate that the administrator is able to specify the action.

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.

Test 1: The evaluator connected the TOE to a test server, and the evaluator had a valid revocation server set up. The TOE is able to query the OCSP revocation status of the peer's certificate (which is valid and unrevoked), and the connection succeeded. The evaluator then connected the TOE to a test server with a test setup in which the TOE was unable to reach the revocation server. The TOE attempts to connect to the unreachable revocation server, but the connection fails, so the TOE rejects the connection attempt to the test server.

Test 2: The TOE checks the peer's certificate validity before it checks the certificate's revocation status. Thus, the TOE is able to detect and reject an invalid certificate before moving on to check its revocation.

2.5 Security management (FMT)

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

2.5.1.1 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 6.1 of the TSS, FMT_CFG_EXT.1 states that the TOE is not installed with any preset default credentials. Users can only access files that are associated to the installation that user performed.

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.

This conditional test is not applicable; the TOE does not use any default credentials.

2.5.1.2 ASPP14:FMT_CFG_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

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

Platforms: Android....

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

Platforms: Microsoft Windows....

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

Platforms: Apple iOS....

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

Platforms: Linux....

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

Platforms: Oracle Solaris....

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

Platforms: Apple macOS....

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

The evaluator used the ProcMon tool to derive a list of files and directories that were written by the TOE during its installation. The evaluator then used the Access Check tool on the resulting file list and found that the tool reported that all items in this list had appropriate R/W permissions for standard Windows users. Standard (non-administrative) windows users had no write access to these files/directories.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.5.2 Supported Configuration Mechanism - per TD0747 & TD0964 (ASPP14:FMT_MEC_EXT.1)

2.5.2.1 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.

The Cisco Jabber TOE reads its configuration stored remotely on the SIP Server.

According to the Application Note, configuration options that are stored remotely are not subject to this requirement. Therefore, this requirement is trivially satisfied.

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/ or https://learn.microsoft.com/en-us/previous-versions/dotnet/netframework-4.0/zzdt0e7f(v=vs.100) or https://learn.microsoft.com/en-us/aspnet/core/fundamentals/configuration/ or https://learn.microsoft.com/en-us/aspnet/core/host-and-deploy/iis/web-config for storing application specific (whether application-wide or user-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 the Windows Registry or C:directory. (TD0964 applied, supersedes TD0893)

Platforms: Apple iOS....

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

Platforms: Linux....

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

Platforms: Oracle Solaris....

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

Platforms: Apple macOS....

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

If ' implement functionality to encrypt and store configuration options as defined by FDP_PRT_EXT.1 in the PP-Module for File Encryption' is selected, for all configuration options listed in the TSS as being stored and protected

using encryption, the evaluator shall examine the contents of the configuration option storage (identified in the TSS) to determine that the options have been encrypted.

The TSS states that the TOE reads its configuration remotely from the configured SIP server. The application note states that configurations stored remotely are not subject to this requirement. Therefore, this test is not applicable.

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

2.5.3.1 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.

The ST does not claim that the TOE can perform any management functions that are identified by the base ASPP. Therefore, there is no additional information that is necessary for the AGD.

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

The Security Target does not claim any management functions for the TOE for the ASPP iteration of this SFR.

2.5.4 Specification of Management Functions (VVoIP Communications) (VVoIPAS10:FMT_SMF.1/VVoIP)

2.5.4.1 VVoIPAS10:FMT_SMF.1.1/VVoIP

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 provides a description of the TOE initial configuration and describes the ability of the TSF to manage the functions that are defined in the SFR, including how each function is managed (e.g. manually configured, applied via downloaded configuration file).

Section 6.1, FMT_SMF.1/VVoIP in the ST states that the initial configuration of the TOE is in an unregistered state. The TSF provides the ability to register the TOE to an Enterprise Session Controller (ESC) and to specify the termination period for idle calls. The registration of the TOE to the ESC is a manual process which can be performed by the TOE once the Operational Environment is appropriately configured by following the configuration steps identified in the AGD.

Section 6.1, FTA_SSL.3/Media in the ST states that the Administrator can specify a remote call disconnect timer parameter in the jabber-config.xml file. If the "CommonCriteriaEndCallTimeout" parameter is specified, the administrator can set a value from 1 to 480 minutes. By default, Jabber will automatically terminate the voice/video media transmission to the remote peer if it determines it is no longer receiving SRTP/SDES traffic after 300 seconds (5 minutes) when operating in CC mode.

Component Guidance Assurance Activities: The evaluator shall verify that the operational guidance provides instructions on configuring any functionality specifically related to VVoIP.

The Security Target claims the following VVOIP management functions:

Ability to register the TOE to an ESC manually;

- Section "Secure Management" of the AGD (specifically, sub-section "SIP Connections and Protocols") provides instructions on configuring the TOE to register to an ESC.

Ability to configure the termination period for idle calls.

- This configuration is addressed by the AGD response for VVOIPAS10:FTA_SSL.3/Media. See that response for details.

Component Testing Assurance Activities: The evaluator shall perform the following tests, depending on the selections made:

Test 1 (conditional: 'register the TOE to an ESC [manually]' is selected):

1. On the TOE, input IP address, gateway address, and subnet mask.

2. If the operational environment is deployed in a manner such that the configuration server and ESC are two distinct servers, input the addresses for each; otherwise, input the ESC address.

3. Save configuration.

4. Verify the TOE registers to the ESC.

Test 2 (conditional: 'register the TOE to an ESC [via DHCP server]' is selected):

1. On the TOE, input the DHCP server address.

2. Save configuration.

3. Verify by traffic capture that the TOE receives all needed IP addresses.

4. Verify by examining the IP address on the TOE.

5. Verify the TOE registers to the ESC.

Test EAs to verify that the TOE can act as a VVoIP call control server when using P2P (if it is selected) are performed as part of testing for FDP_IFF.1/CallControl.

Test EAs to verify the configuration of audit behavior are performed as part of testing for FAU_STG_EXT.1.

Test EAs to verify the modification of transmission of audit data to an external IT entity are performed as part of testing for FAU_STG_EXT.1.

Test EAs to verify that the idle call termination period can be specified are performed as part of testing for FTA_SSL.3/Media, specifically Test 2 and Test 3.

Test EAs to verify that the vocoder can be specified are performed as part of testing for FCO_VOC_EXT.1.

Note that the conditional Test 1 is applicable, and the conditional Test 2 is not applicable based on the Security Target selections. The following addresses Test 1.

The evaluator opened the TOE application and input the network values outlined in the assurance activity. The ESC is not distinct from the configuration server. The evaluator saved this configuration, then input the username and password that was configured on the ESC to be associated with the TOE client. The credentials were accepted by the ESC, and the evaluator was able to follow the prompt provided by the TOE application to complete the registration process. The evaluator observed that the TOE application reported that registration had been completed.

Other EAs for management are satisfied by the SFRs as described in the Assurance Activity.

2.6 Privacy (FPR)

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

2.6.1.1 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 6.1 of the ST, FPR_ANO_EXT.1 states that the TOE does not transmit PII.

Component Guidance Assurance Activities: None Defined

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

This test is not applicable because the TOE does not transmit PII according to the Security Target claims.

2.7 Protection of the TSF (FPT)

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

2.7.1.1 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 6.1 of the ST, FPT_AEX_EXT.1 states that the compiler flag used to enable ASLR when the Cisco Jabber TOE is compiled is /DYNAMICBASE.

The compiler flag used to enable stack-based buffer overflow protection in the Cisco Jabber TOE is /GS.

This indicates that ASLR is enabled. No explicit-mapped exceptions are claimed.

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.

The evaluator used two instances of the TOE application running on different Windows systems. The evaluator used the Microsoft tool VNMap to view and save off the memory addresses of the application. The evaluator compared these memory addresses from both TOE installations on Windows and found that none of memory mapping locations were shared.

The evaluator also executed the Microsoft BinScope tool to check the TOE application for the presence of the /DB (DYNAMICBASE) flag. The result reported from the tool was that the TOE application was compiled with the DB flag, which enables ASLR protection.

2.7.1.2 ASPP14:FPT_AEX_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

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

Platforms: Android....

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

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

o mprotect is never invoked.

Platforms: Microsoft Windows....

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

Platforms: Apple iOS....

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

Platforms: Linux....

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

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

o mprotect is never invoked with the PROT_EXEC permission.

Platforms: Oracle Solaris....

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

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

o mprotect is never invoked with the PROT_EXEC permission.

Platforms: Apple macOS....

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

The evaluator executed the binscope tool with the /NXCheck option. The TOE application successfully passes the check, which means that the executable has DEP protections.

2.7.1.3 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://learn.microsoft.com/en-us/microsoft-365/security/defender-endpoint/enable-exploit-protection?view=o365-worldwide. (TD0823 applied)

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

Platforms: Apple iOS....

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

Platforms: Linux....

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

Platforms: Oracle Solaris....

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

Platforms: Apple macOS....

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

The evaluator checked the Windows OS Platform settings for the configuration of Windows Defender Exploit Guard.

The evaluator ensure that Control Flow Guard (CFG), Randomize memory allocations (Bottom-Up ASLR), Data Execution Prevention (DEP), Export address filtering (EAF) and Import address filtering (IAF) were enabled.

The TOE is Windows 11 and therefore does not support the EMET.

With these settings, the evaluator restarted the TOE application, then observed automatic connection to the ESC, and successfully placed a call. This verified that the TOE was still able to operate normally.

2.7.1.4 ASPP14:FPT_AEX_EXT.1.4

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

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

Platforms: Android....

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

Platforms: Microsoft Windows....

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

Platforms: Apple iOS....

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

Platforms: Linux....

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

Platforms: Oracle Solaris....

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

Platforms: Apple macOS....

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

The evaluator started the TOE application and mimicked normal usage. The evaluator then used the ProcMon tool to identify which locations were written to during TOE operation. The evaluator examined any user-writable file locations and subdirectories within this list for any instance of TOE executable files and found no such instance.

2.7.1.5 ASPP14:FPT_AEX_EXT.1.5

TSS Assurance Activities: (Conditional: The PE or ELF automated tests fail) The evaluator shall ensure that the TSS describes the stack-based buffer overflow compiler flags. (TD0815 applied)

Section 6.1 of the ST, FPT_AEX_EXT.1 states that the compiler flag used to enable stack-based buffer overflow protection in the Cisco Jabber TOE is /GS.

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, BinSkim, 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.

If these automated tests fail, the evaluator shall perform the above, conditional TSS activity.

(TD0815 applied)

The evaluator reviewed the TSS and found that it attests that the /GS flag was used during compilation. The evaluator also used the tool System Informer to determine that the TOE was appropriately implementing stack protection.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

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

2.7.2.1 ASPP14:FPT_API_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.1 of the ST, FPT_API_EXT.1 lists the Windows Platform APIs that the TOE application leverages.

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.

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

2.7.3.1 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 6.1 of the ST, FPT_IDV_EXT.1 states that the Cisco Jabber TOE uses a sequence-based versioning control system. The application uses the "major.minor. maintenance.build" format for versioning control. For example: 15.0.4.56553.

- Major (15 in the example above) designates a release where significant new features are added.
- Minor (0 in the example above) designates a release where minor new features are added.
- Maintenance Release (4 in the example above).
- Build (56553 in the example above) are typically patches for vulnerabilities or bugs.

Component Guidance Assurance Activities: None Defined

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

The evaluator followed the AGD to query the current version of the TOE software. The TOE reported that it was running Version 15.2, which matches the documented and installed version.

The TOE does not claim the use of SWID tags.

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

2.7.4.1 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 examined the TOE installation directory and compiled a list of each dynamic library that was found. From this, the evaluator derived a list of libraries and compared it to the list presented in the assignment within the ST. The evaluator found that each library that was packaged with (or employed by) the TOE application were covered by the libraries identified within the assignment.

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

2.7.5.1 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 "Product Updates" of the AGD contains instructions to query the current version of the TOE and how to check for updates.

The same section also states that when software updates are made available by Cisco, an administrator can obtain, verify the integrity of, and install those updates. The updates are a new version of the TOE and can be downloaded from Cisco.com. The client device platform automatically verifies the digital signature of the TOE software to ensure it has not been modified from the originals distributed by Cisco Systems, Inc.

Section "TOE Installation" and subsection "Install Jabber" contains instructions for securely accepting the TOE and any subsequent TOE updates. The instructions for installation are also the instructions for updating when a new version of the TOE is received.

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

The evaluator followed the AGD instructions to check for updates. The TOE reported that it checked for updates and found that none were available.

The evaluator concludes that the TOE is able to check for updates without error. The requirement is met since the TOE correctly reported that no update was available.

2.7.5.2 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 "Product Updates" of the AGD contains instructions to query the current version of the TOE and how to check for updates.

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 followed the AGD to query the current version of the TOE software. The TOE reported that it was running Version 15.2, which matches the documented and installed version.

2.7.5.3 ASPP14:FPT_TUD_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

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

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

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

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

The evaluator took hashes of all executable files within the TOE install directory. The evaluator then exercised all of the TOE features that are identified by the Security Target. The evaluator then took the hashes of these executables again and found that they were identical.

Therefore, the evaluator concludes that the TOE executable files do not change as a result of exercising TOE functionality.

2.7.5.4 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.

See the TSS response for ASPP14:FPT_TUD_EXT.2.3 which covers the signing of the application by an authorized source.

Section 6.1 of the ST, FPT_TUD_EXT.1 indicates that updates are made available by the vendor. Section "TOE Installation" and subsection "Install Jabber" in the AGD contains instructions for securely accepting the TOE and any subsequent TOE updates.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.7.5.5 ASPP14:FPT_TUD_EXT.1.5

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

Section 6.1 of the ST, FPT_TUD_EXT.1 states that when TOE updates are made available by Cisco, an administrator can obtain and install the update. The authorized source for the digitally signed updates is "Cisco Systems, Inc". In a production environment, distribution to end users is typically achieved by using a Mobile Device Management (MDM) solution. MDM solutions allow Administrators to deploy the TOE software in mass while maintaining control of software version, mode of operation, and feature selection.

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.

"As an additional package" is selected. Therefore, see tests for FPT_TUD_EXT.2.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.7.6 Trusted Update - per TD0730 (VVoIPAS10:FPT_TUD_EXT.1)

2.7.6.1 VVoIPAS10:FPT_TUD_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: There is no change to the EAs specified for this SFR in the App PP. Note however that the following additional configuration steps may be necessary when relying on a call control server or dedicated file server in order for this testing to be performed:

- The evaluator shall deploy a call control server or dedicated file server in the TOE's operational environment

- The evaluator shall load valid and invalid candidate updates to the call control server or dedicated file server

- The evaluator shall configure the TOE or its operational environment (whichever is selected to implement the security functions) to use the call control server or dedicated file server as its source for software/firmware updates

(TD0730 applied)

There is no change to the EAs specified for this SFR in the App PP. See ASPP14:FPT_TUD_EXT.* SFRs for details.

2.7.7 Integrity for Installation and Update - per TD0628 (ASPP14:FPT_TUD_EXT.2)

2.7.7.1 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 evaluator examined the TOE installer file provided by vendor and found that the Windows Platform reported that the installer was in the MSI format. This meets the testing requirement.

2.7.7.2 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)

Prior to installing the TOE, the evaluator recorded the file path of every file on the Windows 11 filesystem. The evaluator then installed and ran the application. The evaluator then uninstalled the TOE and recorded these paths again.

The evaluator then analyzed the differences between the two lists, the evaluator found that multiple files had been created during the install, run, and uninstall process. However, all of the files that remained after uninstallation were some form of configuration, output, or logs (or were otherwise files that changed as a result of normal functionality of the Windows 11 platform).

In conclusion, the evaluator observed that only configuration, output, and log files had been added to the filesystem. The TOE did not add any executable files that were not removed with the uninstall process.

2.7.7.3 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 6.1 of the TSS, FPT_TUD_EXT.2 states that TOE software is digitally signed using RSA 4096-bit key and SHA256 hash, which is in line with the claims made for FCS_COP.1/Hash and FCS_COP.1/Sig. The platform performs verification and performs an install/upgrade/update if verification is successful. The authorized source for the digitally signed updates is "Cisco Systems, Inc".

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

2.8 TOE ACCESS (FTA)

2.8.1 /MEDIA TSF-INITIATED TERMINATION (MEDIA CHANNEL) (VVOIPAS10:FTA_SSL.3/MEDIA)

2.8.1.1 VVOIPAS10:FTA_SSL.3.1/MEDIA

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 specifies whether idle calls are terminated by default after a certain amount of time or by an administrator-configurable interval.

Section 6.1 of the ST, FTA_SSL.3/Media states that the Administrator can specify a remote call disconnect timer parameter in the jabber-config.xml file. If the "CommonCriteriaEndCallTimeout" parameter is specified, the administrator can set a value from 1 to 480 minutes. By default, Jabber will automatically terminate the

voice/video media transmission to the remote peer if it determines it is no longer receiving SRTP/SDES traffic after 300 seconds (5 minutes) when operating in CC mode.

Component Guidance Assurance Activities: If the idle call timeout period is administrator-configurable, the evaluator shall verify that the operational guidance includes instructions for how to configure this, as well as what the minimum and maximum allowed values are.

Section "CUCM UC Service" of the AGD explains how to configure the TOE's idle call timeout period. It indicates that the minimum value is 60 seconds (1 minute) and that the maximum value is 28800 seconds (480 minutes). These values are consistent with the corresponding TSS description.

Component Testing Assurance Activities: The evaluator shall set up a test environment that contains the TOE, a call control server (which may be the TOE itself), a configuration server (if used to communicate configuration changes to idle timeout period), a network switch, a traffic capture tool, and a second VVoIP endpoint.

The evaluator shall then perform the following tests:

Test 1:

1. Deploy the TOE in a default configuration (i.e. without any administrative override applied to the idle timeout value).

2. Ensure the TOE is using P2P or register the TOE with the call control server and verify that it is registered by viewing its status on the ESC and capturing the call control path traffic.

3. Ensure the second VVoIP endpoint is using P2P or register the second VVoIP endpoint with the call control server and verify that it is registered by viewing its status on the call control server and capturing the call control path traffic.

4. Use the TOE to dial the second VVoIP endpoint and establish a call. Verify the call was established by holding a conversation between the two peers and capturing the streaming media traffic that is transmitted between them.

5. Power down the second VVoIP endpoint while the call is active. Observe that the TOE stops transmitting media after the default period of time specified in the ST.

Test 2 (conditional:'an administrator-configurable interval' is selected in FTA_SSL.3.1/Media):

1. Deploy the TOE in a default configuration (i.e. without any administrative override applied to the idle timeout value).

2. Ensure the TOE is using P2P or register the TOE with the call control server and verify that it is registered by viewing its status on the call control server and capturing the call control path traffic.

3. Configure the TOE's idle timeout period for the shortest period of time that is supported.

4. Ensure the second VVoIP endpoint is using P2P or register the second VVoIP endpoint with the call control server and verify that it is registered by viewing its status on the call control server and capturing the call control path traffic.

5. Use the TOE to dial the second VVoIP endpoint and establish a call. Verify the call was established by holding a conversation between the two peers and capturing the streaming media traffic that is transmitted between them.

6. Power down the second VVoIP endpoint while the call is active. Observe that the TOE stops transmitting media after the period of time configured in Step 3.

Test 3 (conditional: 'an administrator-configurable interval' is selected in FTA_SSL.3.1/Media):

Repeat Test 2 but in Step 3, configure the idle timeout value to be the longest period of time that is supported as opposed to the shortest.

Test 1:

Test 1 was performed using the default TOE timeout configuration as required. The default value is 5 minutes. The evaluator verified that the TOE and a peer endpoint were registered to the ESC, and were using the correct call control path. The evaluator first began a packet capture capturing the traffic between the TOE and a peer endpoint, as well as between the TOE and the call control server they were registered to.

The evaluator then placed a call to the peer endpoint, held a brief conversation, then powered down the peer endpoint. After 5 minutes, the TOE automatically ended the call due to inactivity. Through examining the packet capture, the evaluator determined that the TOE correctly stopped transmitting streaming media after the 5-minute idle time period had elapsed.

Test 2:

The TOE claims to have an administratively configurable interval, which is set on the ESC. The evaluator followed the test procedure identified in Test 1, but this time used the minimum possible value which is 1 minute, and the greatest possible time value which is 8 hours. The evaluator observed that the TOE automatically ended the call due to inactivity after the configured value (1 minute or 8 hours). Through examining the packet capture, the evaluator also determined that the TOE correctly stopped transmitting streaming media after the configured idle time period had elapsed.

2.9 TRUSTED PATH/CHANNELS (FTP)

2.9.1 PROTECTION OF DATA IN TRANSIT - PER TD0743 (ASPP14:FTP_DIT_EXT.1)

2.9.1.1 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 6.1 of the ST, FTP_DIT_EXT.1 states that the TOE uses TLS and SRTP to encrypt transmitted sensitive data. This TLS/SRTP functionality is implemented, and therefore no information about the platform is required in the TSS.

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.

Tests 1 and 2: Through examining a packet capture of TOE activity, the evaluator determined that sensitive data to remote systems such as the ESC in the operational environment was protected by TLS. The evaluator observed that no sensitive data was transmitted in cleartext to remote systems.

Test 3: While capturing traffic from the TOE, the evaluator provided credentials to authenticate and register to the ESC. The evaluator searched this packet capture for any instance of these credentials in cleartext, finding none. Therefore, the evaluator concludes that the TOE does not transmit plaintext data, including credentials, and protects traffic using TLS.

Test 4 (Android & IOS platforms): Not applicable. The TOE is not Android or iOS.

2.9.2 Protection of Data in Transit - per TD0785 (VVoIPAS10:FTP_DIT_EXT.1)

2.9.2.1 VVoIPAS10:FTP_DIT_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: None Defined

2.9.3 Inter-TSF Trusted Channel (Signaling Channel) (VVoIPAS10:FTP_ITC.1/Control)

2.9.3.1 VVoIPAS10:FTP_ITC.1.1/Control

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 describes the ability of the TOE to use SIP and/or H.323 with TLS.

Section 6.1 of the ST, FTP_ITC.1/Control states that each time the TOE initiates or receives a VVoIP call, TLS is used to protect the SIP session management communication between itself and the ESC. Communication between the TOE and the remote VVoIP endpoint is protected with SRTP.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The vendor shall provide to the evaluator application layer configuration settings for all secure communication mechanisms specified by the FTP_ITC.1/Control requirement. This information should be sufficiently detailed to allow the evaluator to determine the application layer timeout settings for each cryptographic protocol. There is no expectation that this information must be recorded in any public-facing document or report.

The evaluator shall perform the following tests:

Test 1: The evaluators shall ensure that communications using each protocol with a call control server is tested during the course of the evaluation, setting up the connections as described in the guidance documentation and ensuring that communication is successful.

Test 2: For each protocol that the TOE can initiate as defined in the requirement, the evaluator shall follow the guidance documentation to ensure that in fact the communication channel can be initiated from the TOE.

Test 3: The evaluator shall ensure, for each communication channel with an authorized IT entity, the channel data is not sent in plaintext.

Test 4: Objective: The objective of this test is to ensure that the TOE reacts appropriately to any connection outage or interruption of the route to the external IT entities.

The evaluator shall, for a connection to a call control server using each claimed protocol, physically interrupt the connection to the call control server for the following durations: i) a duration that exceeds the TOE's application layer timeout setting, ii) a duration shorter than the application layer timeout but of sufficient length to interrupt the MAC layer.

The evaluator shall ensure that, when the physical connectivity is restored, communications are appropriately protected and no TSF data is sent in plaintext.

In the case where the TOE is able to detect when the cable is removed from the device, another physical network device (e.g. a core switch) shall be used to interrupt the connection between the TOE and the call control server. The interruption shall not be performed at the virtual node (e.g. virtual switch) and must be physical in nature.

Further EAs are associated with the specific protocols.

Test 1: As shown in the test results for ASPP14:FDP_NET_EXT.1, the evaluator confirmed that all communications from the TOE to the ESC and to another VoIP endpoint is successful. The communications are encrypted accordingly.

Test 2: The test results for ASPP14:FDP_NET_EXT.1 show that the communications are TOE initiated.

Test 3: The test results for ASPP14:FDP_NET_EXT.1 show that the data is encrypted and not transmitted in plaintext.

Test 4:

The evaluator captured traffic between the TOE and call control server. Since the Windows platform is able to detect when an interface is disconnected, the network was set up in such a way that a core switch was utilized, which was physically powered off to achieve the disconnection.

First, the evaluator tested the MAC layer disconnection timeout. For this, the evaluator disrupted the connection between the TOE and the call control server for approximately long enough to cause the MAC layer disconnect. After restoring the connection, the TOE was observed to continue to use the active TLS connection to the call control server and properly resumed encrypted communication without sending any data in plaintext.

Next, the evaluator tested the APP layer disconnection timeout. For this, the evaluator disrupted the connection between the TOE and the call control server for long enough to achieve the application layer timeout. After restoring the connection, the TOE was observed to initiate a new TLS connection to the call

control server, the properly resumed encrypted communication to the call control server without sending any data in plaintext.

In conclusion, regardless of whether the TOE connection to the call control server is disrupted for the MAC or Application layer timeout, the TOE is able to properly resume communications on the encrypted TLS channel without exposing any TSF data in plaintext.

2.9.4 Inter-TSF Trusted Channel (Media Channel) (VVoIPAS10:FTP_ITC.1/Media)

2.9.4.1 VVoIPAS10:FTP_ITC.1.1/Media

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 trusted channel will use SRTP or H.323/H.235 in accordance with the selections made in the SFR.

Section 6.1 of the ST, FTP_ITC.1/Control states that each time the TOE initiates or receives a VVoIP call, TLS is used to protect the SIP session management communication between itself and the ESC. Communication between the TOE and the remote VVoIP endpoint is protected with SRTP.

This is consistent with the SRTP selection made in FTP_ITC.1/Media.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The vendor shall provide to the evaluator application layer configuration settings for all secure communication mechanisms specified by the FTP_ITC.1/Media requirement. This information should be sufficiently detailed to allow the evaluator to determine the application layer timeout settings for each cryptographic protocol. There is no expectation that this information must be recorded in any public-facing document or report.

The evaluator shall perform the following tests:

Test 1: The evaluators shall ensure that communications using each protocol with another VVoIP endpoint is tested during the course of the evaluation, setting up the connections as described in the guidance documentation and ensuring that communication is successful.

Test 2: For each protocol that the TOE can initiate as defined in the requirement, the evaluator shall follow the guidance documentation to ensure that in fact the communication channel can be initiated from the TOE.

Test 3: The evaluator shall ensure, for each communication channel with an authorized IT entity, the channel data is not sent in plaintext.

Test 4: Objective: The objective of this test is to ensure that the TOE reacts appropriately to any connection outage or interruption of the route to the external IT entities.

The evaluator shall, for a connection to another VVoIP endpoint device using each claimed protocol, physically interrupt the connection to that remote endpoint for the following durations: i) a duration that exceeds the TOE's application layer timeout setting, ii) a duration shorter than the application layer timeout but of sufficient length to interrupt the MAC layer.

The evaluator shall ensure that, when the physical connectivity is restored, communications are appropriately protected and no TSF data is sent in plaintext.

In the case where the TOE is able to detect when the cable is removed from the device, another physical network device (e.g. a core switch) shall be used to interrupt the connection between the TOE and the remote endpoint. The interruption shall not be performed at the virtual node (e.g. virtual switch) and must be physical in nature.

Test 1: As shown in the test results for ASPP14:FDP_NET_EXT.1, the evaluator confirmed that all communications from the TOE to the ESC and to another VoIP endpoint is successful. The communications are encrypted accordingly.

Test 2: The test results for ASPP14:FDP_NET_EXT.1 show that the communications are TOE initiated.

Test 3: The test results for ASPP14:FDP_NET_EXT.1 show that the data is encrypted and not transmitted in plaintext.

Test 4: The evaluator captured traffic between the TOE and call control server, as well as the peer endpoint. Since the Windows platform is able to detect when an interface is disconnected, the network was set up in such a way that a core switch was used which was physically powered off to achieve the disconnection.

First, the evaluator tested the MAC layer disconnection timeout. For this, the evaluator disrupted the connection between the TOE and the second VVOIP peer endpoint for long enough to reach the MAC layer timeout. After restoring the connection, the TOE was observed to continue to use the active TLS connection to the call control server and properly resumed encrypted SRTP communication to the peer endpoint without sending any data in plaintext.

Next, the evaluator tested the APP layer disconnection timeout. For this, the evaluator disconnect the connection between the TOE and the peer endpoint for long enough to reach the application layer timeout. After the connection was restored, the TOE was observed to initiate a new TLS connection to the call control server, and properly resumed encrypted SRTP communication to the peer endpoint.

In conclusion, regardless of whether the TOE is disconnected for the MAC or Application layer timeout, it is able to properly resume communications on the encrypted SRTP channel without exposing any TSF data in plaintext.

3. Protection Profile SAR Assurance Activities

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

3.1 Development (ADV)

3.1.1 Basic Functional Specification (ADV_FSP.1)

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

3.2 Guidance documents (AGD)

3.2.1 Operational User Guidance (AGD_OPE.1)

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

Cryptographic functions are provided by the TOE as described in the TSS. Section "SIP Connections and Protocols" in the AGD states that there is no direct admin or user interaction on the TOE to configure or set the SRTP channel and no user configuration is required for the cryptographic features of the TOE. Additionally, in the evaluated

configuration, there is no mechanism for a user to cause the TOE to use a different cryptographic engine than the one that was evaluated, and therefore such a warning to the Administrator is not necessary in the AGD.

See the AGD response for ASPP14:FPT_TUD_EXT.1.1 which addresses the platform's ability to verify the digital signature. This response also indicates the sections of the AGD that addresses the steps for obtaining and provisioning the update, as well as initiating the update process and the indication of success.

The AGD includes the section "Excluded and Functionality Not Covered" which clearly indicates which functionality of the TOE is excluded from the scope of the ASPP/PKGTLS/VVOIP protection profiles and therefore not covered by the evaluation activities.

3.2.2 Preparative Procedures (AGD_PRE.1)

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

The only platform claimed is Windows 11 which is consistent in the AGD and ST. The evaluator found that the instructions presented in the AGD were sufficient to address any necessary configuration of Windows 11 to support the TOE.

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.

Throughout the ST and AGD, the TOE is uniquely identified as Cisco Jabber 15.2. The evaluator found this to be consistent with the TOE samples received from the vendor (e.g., FPT_TUD testing). The evaluator examined the vendor's page for the product and found that the TOE was easily distinguishable.

https://www.cisco.com/c/en/us/products/unified-communications/jabber/index.html

3.3.2 TOE CM Coverage (ALC_CMS.1)

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

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

The evaluator verified that the ST, TOE and AGD are all labeled with identifiable versions. The evaluator checked the TOE version during testing by examining the TOE software during testing.

The AGD 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 6.1 of the ST, ALC_TSU_EXT.1 states that the TOE has specific versions that can be queried by a user. A TOE update is not a patch applied to the existing TOE; it is a new version of the TOE. When TOE updates are made available by Cisco, an administrator can obtain and install the update. TOE software is digitally signed using RSA 4096-bit key and SHA256 hash, which is in line with the claims made for FCS_COP.1/Hash and FCS_COP.1/Sig. The platform performs verification and performs an install/upgrade/update if verification is successful. The authorized source for the digitally signed updates is "Cisco Systems, Inc". In a production environment, distribution to end users is typically achieved by use of a Mobile Device Management (MDM) solution. MDM solutions allow Administrators to deploy the TOE software in mass while maintaining control of software version, mode of operation, and feature selection.

All Cisco communications relating to security issues are handled by the Cisco Product Security Incident Response Team (PSIRT). Cisco aims to provide fixes in 30 days, but depending on the timing, it may be greater than 30 days though not more than 60 days for most security issues. Fixes may be delayed longer for low-risk security issues. Updates are then made available at Cisco Software Central at: https://software.cisco.com.

Customers can subscribe to the Cisco Notification Service to receive important information regarding product updates. Full information is provided in the Cisco Security Vulnerability Policy available at https://tools.cisco.com/security/center/resources/security_vulnerability_policy.html

Customers may securely report security issues to Cisco Support. For details on engaging Cisco Support, please reference Annex E: Obtaining Documentation and Submitting a Service Request.

3.4 Tests (ATE)

3.4.1 Independent Testing - Conformance (ATE_IND.1)

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

While it is not necessary to have one test case per test listed in an Assurance Activity, the evaluator must document in the test plan that each applicable testing requirement in the ST is covered. The test plan identifies the platforms to be tested, and for those platforms not included in the test plan but included in the ST, the test plan provides a justification for not testing the platforms. This justification must address the differences between the tested platforms and the untested platforms, and make an argument that the differences do not affect the testing

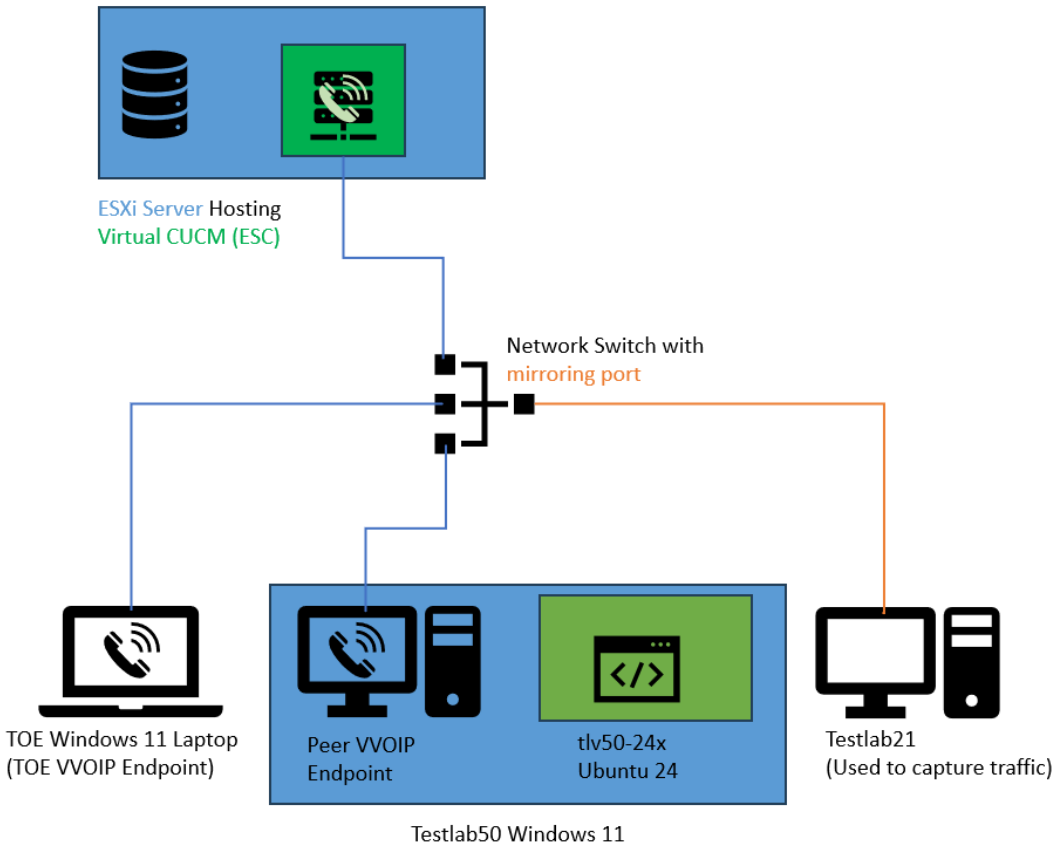
to be performed. It is not sufficient to merely assert that the differences have no affect; rationale must be provided. If all platforms claimed in the ST are tested, then no rationale is necessary. The test plan describes the composition of each platform to be tested, and any setup that is necessary beyond what is contained in the AGD documentation. It should be noted that the evaluator is expected to follow the AGD documentation for installation and setup of each platform either as part of a test or as a standard pre-test condition. This may include special test drivers or tools. For each driver or tool, an argument (not just an assertion) should be provided that the driver or tool will not adversely affect the performance of the functionality by the TOE and its platform.

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

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

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

The following diagram depicts the evaluator’s test environment:



TOE Platforms:

- Windows 11 device with Intel Core i5-1135G7 (Tiger Lake) CPU

Supporting Products:

- Cisco Unified Communications Manager (CUCM) version 15SU2
- Ubuntu 24 6.8.0-60-generic
- Microsoft Windows 11 Pro 10.0.26100

Supporting Software:

- SIPp v3.7.3-36-ge3716c5
- Wireshark version 4.47
- Nmap version 7.94
- Dependency Walker v2.2

- ProcMon (Process Monitor) Version 4.01
- AccessChk (Access Check) Version 6.15
- VMMap v3.4
- WinDbg v1.2504.15001.0
- BinSkim 4.4.1
- System informer 3.2.25011.2031
- Microsoft Binscope 2014 Version 7.0.7000.0
- WinMerge version 2.16.14.0

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 MITRE CVE Database, National Vulnerability Database, and CVE details (https://www.cve.org/, https://web.nvd.nist.gov/vuln/search, and https://www.cvedetails.com/vulnerability-search.php, ref CVE) and Known Vulnerability Exploit Catalog (https://www.cisa.gov/known-exploited-vulnerabilities-catalog, ref KEV) on 06/25/2026 (from 4/1/2025) with the following search terms: "Cisco Jabber", "Cisco SIP", "Cisco SRTP", "Cisco TCP", "Windows 11", "Intel Core i5-1135G7", "Tiger Lake", "nghttp2", "gssapi32", "tiny-xml", "leveldb", "plantronics software", "chromium", "openvino toolkit", "krbcc32", "kfwlogon", "ippsp legacy", "iintel integrated performance primitives", "signal processing legacy", "angle", "ippsc legacy iintel integrated performance primitives", "speech codecs legacy", "visual-studio-runtime", "microsoft windows operating system", "openjpeg", "libusb", "leashw32", "microsoft edge embedded browser webview loader", "krb5_32", "kerberos", "skia", "json-cpp", "gloox", "opencore-amr", "plantronics jabber plugin", "debugging tools for windows", "gstreamer", "zlib", "free-type", "threading-building-blocks", "oneapi threading building blocks", "sqlite", "io-warrior-sdk", "sigc++", "libxml2", "glib", "re2", "zlib", "cyrus-sasl", "libvorbis", "pdfium", "tidy", "curl", "boringssl", "openssl", "libjpeg-turbo", "direct3d", "protobuf", "open-ldap", "expat", "gsoap", "libcxx", "cef", "json-

c", "libxslt", "pcre", "jansson", "libvpx", "sql-cipher", "xpprof32", "blink", "k5sprt32", "boost", "libjpeg", "libphonenumber", "opus", "libpng", "safestring", "libwebp", "glew", "hunspell", "rapidxml", "speexdsp", "comerr32", "CertVerifyCertificateChainPolicy", "CertGetCertificateChain", "CryptAcquireContext", "CryptCreateHash", "CryptHashData", "CryptGetHashParam", "BCryptGenRandom", "BCrypt", "international components for unicode", "Google v8", "Apache Portable Runtime", and "OpenType Sanitizer".

The evaluator searched the vulnerability sites above using the keywords shown above. At that time all issues found were considered and a resolution determined for each issue. The evaluator determined that there are no open public issues.

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

On the same day as the initial vulnerability search above (to ensure the most recent definitions), the evaluator used Windows Defender on the Windows 11 platform to scan the TOE application files. The evaluator found that none were flagged by the virus scanner.