www.GossamerSec.com

Assurance Activity Report for DataSoft Secure Tactical VPN Client for Android

Version 0.4
08/07/23

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

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

Revision History

Revision	Date	Authors	Summary
Version 0.1	06/23/23	Smoley	Initial draft
Version 0.2	07/18/23	Smoley	Updated ST reference
Version 0.3	07/26/23	Smoley	Updated for ECR comments
Version 0.4	08/07/23	Smoley	Updated for ECR comments

The TOE Evaluation was Sponsored by:

DataSoft Corporation
10235 S 51st Street, #115
Phoenix, AZ 85044

Evaluation Personnel:

- Raymond Smoley
- Yoel Fortaleza

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 DataSoft Secure Tactical VPN Client for Android ASPP14/VPNC24 evaluation. This document contains a description of the assurance activities and associated results as performed by the evaluators.

1.1 Device Equivalence

The TOE was specifically tested on those three different versions of Android using the following hardware

Phone	Model	CPU	Kernel	Android OS	VID/Date
Samsung	S20 Tactical Edition	Qualcomm snapdragon 865 (SM8250)	4.19	Android 11	11042/ Archived
Google	Pixel 5	Qualcomm snapdragon 765G (SM7250)	4.19	Android 11	11124/ Archived
Google	Pixel 4a-5G	Qualcomm snapdragon 765G (SM7250)	4.19	Android 12	11239/ 02/28/2022
Google	Pixel 5a-5G	Qualcomm snapdragon 765G (SM7250)	4.19	Android 13	11317/ 01/24/2023

The devices above were identified based on availability for testing; however, these devices were previously evaluated to exhibit the same behavior from a security function standpoint. Since the TOE is distributed as a singular application for all platforms, uses standard Android-supported APIs, and is run through Android's Dalvik Virtual Machine which obfuscates many of the calls to lower-level functions, the evaluation team concludes that any other Android 11, 12 or 13 device can run the TOE in a security-functionally equivalent manner.

1.2 CAVP Certificates

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

The TOE's Secure Tactical VPN Client OpenSSL Cryptographic Library version 3.1.0 possesses the following cryptographic algorithm certificates:

Requirements	Functions	CAVP Cert
	Cryptographic key generation	
ASPP14:FCS_CKM_EXT.1 ASPP14:FCS_CKM.1/AK VPNC24:FCS_CKM.1 VPNC24:FCS_CKM/AK	ECC schemes using 'NIST curves' P-256, P-384	A3718
	Cryptographic key establishment/distribution	
ASPP14:FCS_CKM.2 VPNC24:FCS_CKM.2	Elliptic curve-based key establishment schemes: P-256, P-384	A3718
	IPsec/ESP Encryption/Decryption	
ASPP14:FCS_COP.1/SKC	AES CBC/GCM (128/256 bits)	A3718

Requirements	Functions	CAVP Cert
VPNC24:FCS_COP.1/SKC		
	Cryptographic hashing	
ASPP14:FCS_COP.1/Hash	SHA-256/384/512 (digest size 256/384/512 bits)	A3718
	Keyed-hash message authentication	
ASPP14:FCS_COP.1/KeyedHash	HMAC-SHA-256/384/512 (key and output MAC size 256/384/512)	A3718
	Cryptographic signature services	
ASPP14:FCS_COP.1/Sig	Elliptic Curve Digital Signature Algorithm (ECDSA) with an elliptical curve P-256, P-384	A3718
	Random bit generation	
FCS_RBG_EXT.1	CTR_DRBG (AES-256 bit)	A3718

2. Protection Profile SFR Assurance Activities

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

The following evidence was used to complete the Assurance Activities:

AAR v0.4

- DataSoft Secure Tactical VPN Client for Android Security Target (ST/TSS), Version 0.5, 08/07/23
- DataSoft Secure Tactical VPN Client CC Configuration Guide (AGD), Version 1.1, July 26, 2023

2.1 Cryptographic support (FCS)

2.1.1 Cryptographic Key Generation Services (VPNC24:FCS_CKM.1)

2.1.1.1 VPNC24:FCS_CKM.1.1

TSS Assurance Activities: This SFR is evaluated in conjunction with FCS_CKM.1/AK in the App PP

Guidance Assurance Activities: This SFR is evaluated in conjunction with FCS_CKM.1/AK in the App PP

Testing Assurance Activities: This SFR is evaluated in conjunction with FCS_CKM.1/AK in the App PP

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

2.1.2.1 ASPP14:FCS_CKM.1.1/AK

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

The TOE implements functionality to generate asymmetric cryptographic keys. Section 6.1 of the TSS states that the TOE uses asymmetric keys as the TOE supports DH groups 19 and 20 and thus supports generation of ECC asymmetric keys against NIST curves P-256 and P-384 respectively.

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 3.2 TOE Configuration of the AGD contains information about configuring the TOE with the correct key generation algorithms used for IPsec including limiting the TOE to the correct scheme and key sizes (i.e. P-256 and P-384) for CC and CSfC deployments. Additionally, Section 1.5 Excluded Functionality contains a reference to these sections and notes that outside of these configurations, the TOE requires no additional configuration of cryptographic services to be in a CC-compliant mode. No further key generations schemes are defined in the evaluation.

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

1. Random Primes:

Provable primes

Probable primes

2. Primes with Conditions:

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

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

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

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

n = p*q,

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

GCD(p-1,e) = 1,

GCD(q-1,e) = 1,

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

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

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

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

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

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

Key Generation for Elliptic Curve Cryptography (ECC)

FIPS 186-4 ECC Key Generation Test For each supported NIST curve, i.e., P-256, P384 and P-521, the evaluator shall require the implementation under test (IUT) to generate 10 private/public key pairs. The private key shall be generated using an approved random bit generator (RBG). To determine correctness, the evaluator shall submit the generated key pairs to the public key verification (PKV) function of a known good implementation. FIPS 186-4 Public Key Verification (PKV) Test For each supported NIST curve, i.e., P256, P-384 and P-521, the evaluator shall generate 10 private/public key pairs using the key generation function of a known good implementation and modify five of the public key values so that they are incorrect, leaving five values unchanged (i.e., correct). The evaluator shall obtain in response a set of 10 PASS/FAIL values. Key Generation for Finite-Field Cryptography (FFC) The evaluator shall verify the implementation of the Parameters Generation and the Key Generation for FFC by the TOE using the Parameter Generation and Key Generation test. This test verifies the ability of the TSF to correctly produce values for the field prime p, the cryptographic prime q (dividing p-1), the cryptographic group generator g, and the calculation of the private key x and public key y. The Parameter generation specifies 2 ways (or methods) to generate the cryptographic prime q and the field prime p:

Cryptographic and Field Primes:

Primes q and p shall both be provable primes

Primes q and field prime p shall both be probable primes

and two ways to generate the cryptographic group generator g:

Cryptographic Group Generator:

Generator g constructed through a verifiable process

Generator g constructed through an unverifiable process.

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

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

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

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

g not= 0,1

q divides p-1

g^q mod p = 1

g^x mod p = y

for each FFC parameter set and key pair.

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

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

The TOE has been CAVP tested. Refer to the CAVP certificates identified in Section 1.2 of this document.

2.1.3 Cryptographic Asymmetric Key Generation (VPNC24:FCS_CKM.1/AK)

2.1.3.1 VPNC24:FCS_CKM.1.1/AK

TSS Assurance Activities: This SFR is evaluated in conjunction with FCS_CKM.1/AK in the App PP

Guidance Assurance Activities: This SFR is evaluated in conjunction with FCS_CKM.1/AK in the App PP

Testing Assurance Activities: This SFR is evaluated in conjunction with FCS_CKM.1/AK in the App PP

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

2.1.4.1 ASPP14:FCS_CKM.1.1/SK

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

Section 6.1 of the TSS states that the TOE uses its own OpenSSL library’s DRBG to generate random values as part of IKEv2/CHILD_SA secret key generation. Additionally, the TOE seeds it using the ASPP14 prescribed method of calling the Android platform’s /dev/random interface.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.1.5 VPN Cryptographic Key Generation (IKE) (VPNC24:FCS_CKM.1/VPN)

2.1.5.1 VPNC24:FCS_CKM.1.1/VPN

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 key generation functionality is invoked.

Section 6.1 of the ST states that the TOE invokes Android's keychain functionality to secure manage IKE authentication parameters. Android provides the user with a secure (either Android's Keystore, a Trust-Zone backed key storage tied to the CPU, or Android's StrongBox keystore, a hardware secure element with its own dedicated storage) keychain, in which they can import certificates and private keys. Android provides methods to allow generation and import of a certificate and private key. Android provides the both the KeyPairGenerator and the Android KeyStore methods to allow secure generation of keys. In addition to generation, the TOE's UI presents an interface to Androids System UI to import a certificate (chain) and private key in p12/PFX format, or alternatively, the user can separate load the p12 file through Android's System UI (an MDM Agent or Device Policy Controller can also import p12 certificates as directed by an MDM server). When creating a new VPN profile, the TOE prompts the user to select the certificate/private key they wish the TOE to use during IKE authentication.

Component Guidance Assurance Activities: There are no AGD Assurance Activities for this requirement.

There are no AGD Assurance Activities for this requirement

Component Testing Assurance Activities: If this functionality is implemented by the TSF, refer to the following EAs, depending on the TOE's claimed Base-PP:

- GPOS PP: FCS_CKM.1

- MDF PP: FCS_CKM.1

- App PP: FCS_CKM.1/AK

- MDM PP: FCS_CKM.1

See ASPP14:FCS_CKM.1/AK

2.1.6 Cryptographic Key Establishment (ASPP14:FCS_CKM.2)

2.1.6.1 ASPP14:FCS_CKM.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall ensure that the supported key establishment schemes correspond to the key generation schemes identified in FCS_CKM.1.1/AK. If the ST specifies more than one scheme, the evaluator shall examine the TSS to verify that it identifies the usage for each scheme. (TD0717 applied)

Section 6.1 of the ST states that the TOE uses only ECC key exchange/establishment and the TOE uses it exclusively as part of IKEv2 and ESP negotiation. This is in agreeance with the claims under FCS_CKM.1.1/AK as this claims the TOE implements functionality to generate/establish ECC keys

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 3.2 TOE Configuration of the AGD contains information about configuring the TOE with the correct key establishment algorithms used for IPsec including the note that "The TOE uses only ECC key exchange/establishment and the TOE uses it exclusively as part of IKEv2 and ESP negotiation. As a result, only ECC client credentials should be configured and the TOE should only be used to connect to servers using ECC server credentials."

Additionally, Section 1.5 Excluded Functionality contains a reference to these sections and notes that outside of these configurations, the TOE requires no additional configuration of cryptographic services to be in a CC-compliant mode.

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 Section 1.2 of this document.

2.1.7 Cryptographic Key Establishment (VPNC24:FCS_CKM.2)

2.1.7.1 VPNC24:FCS_CKM.2.1

TSS Assurance Activities: This SFR is evaluated in conjunction with FCS_CKM.2 in the App PP

Guidance Assurance Activities: This SFR is evaluated in conjunction with FCS_CKM.2 in the App PP

Testing Assurance Activities: This SFR is evaluated in conjunction with FCS_CKM.2 in the App PP

2.1.8 Cryptographic Key Generation Services - per TD0717 (ASPP14:FCS_CKM_EXT.1)

2.1.8.1 ASPP14:FCS_CKM_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

The ST claims that the TOE implements functionality to generate cryptographic keys in accordance to ECC schemes.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.1.9 Cryptographic Key Storage - per TD0725 (VPNC24:FCS_CKM_EXT.2)

2.1.9.1 VPNC24:FCS_CKM_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: Regardless of whether this requirement is met by the TOE or the TOE platform, the evaluator will check the TSS to ensure that it lists each persistent secret (credential, secret key) and private key needed to meet the requirements in the ST. For each of these items, the evaluator will confirm that the TSS lists for what purpose it is used, and how it is stored. The evaluator then performs the following actions:

Persistent secrets and private keys manipulated by the platform:

For each platform listed in the ST, the evaluator shall examine the ST of the platform to ensure that the persistent secrets and private keys listed as being stored by the platform in the VPN client ST are identified as being protected in that platform's ST.

Persistent secrets and private keys manipulated by the TOE:

The evaluator reviews the TSS for to determine that it makes a case that, for each item listed as being manipulated by the TOE, it is not written unencrypted to persistent memory, and that the item is stored by the platform.

Section 6.1 of the ST states that the TOE's only persistent secrets are the IKEv2 authentication keypairs/certificates, which the TOE stores in Android's hardware backed Keystore or in Android's strongbox backed Keystore. The TOE does not store any of its other keys persistently, and instead the TOE stores non-persistent or ephemeral keys pertaining to IKEv2 and ESP SAs only in memory.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.1.10 Cryptographic Key Destruction - per TD0725 (VPNC24:FCS_CKM_EXT.4)

2.1.10.1 VPNC24:FCS_CKM_EXT.4.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall ensure that all plaintext secret and private cryptographic keys and CSPs (whether manipulated by the TOE or exclusively by the platform) are identified in the VPN Client ST's TSS, and that they are accounted for by the EAs in this section.

Requirement met by the platform:

The evaluator shall check to ensure the TSS describes each of the secret keys (keys used for symmetric encryption), private keys, and CSPs used to generate key that are not otherwise covered by the FCS_CKM_EXT.4 requirement levied on the TOE.

For each platform listed in the ST, the evaluator shall examine the TSS of the ST of the platform to ensure that each of the secret keys, private keys, and CSPs used to generate key listed above are covered.

Requirement met by the TOE:

The evaluator shall check to ensure the TSS describes when each of the plaintext keys are cleared (e.g., system power off, disconnection of an IPsec connection, when no longer needed by the VPN channel per the protocol); and the type of clearing procedure that is performed (cryptographic erase, overwrite with zeros, overwrite three or more times by a different alternating pattern, overwrite with random pattern, or block erase). If different types of memory are used to store the materials to be protected, the evaluator shall check to ensure that the TSS describes the clearing procedure in terms of the memory in which the data are stored (for example, 'secret keys stored on flash are cleared by overwriting once with zeros, while secret keys stored on the internal persistent storage device are cleared by overwriting three times with a random pattern that is changed before each write').

Section 6.1 of the ST states that the TOE's keys all pertain to IKEv2/IPsec and consist of the persistent IKEv2 authentication certificate/keypairs (for which the TOE relies upon the Android platform for management, including zeroization) and IKEv2 and ESP SA session keys (which the TOE stores only in working memory and clears after use).

Additionally, the same section contains a table with a reference to the IKEv2 Authentication cert/key and IKEv2/ESP Session keys, where they are stored (Android KeyChain/KeyStore or Working RAM respectively) and how they are cleared (User clearing keychain through Android UI or Automatically cleared when IPsec tunnel closed respectively).

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: For each key clearing situation described in the TSS, the evaluator shall repeat the following test.

Test 1: The evaluator shall utilize appropriate combinations of specialized operational environment and development tools (debuggers, simulators, etc.) for the TOE and instrumented TOE builds to test that keys are cleared correctly, including all intermediate copies of the key that may have been created internally by the TOE during normal cryptographic processing with that key.

Cryptographic TOE implementations in software shall be loaded and exercised under a debugger to perform such tests. The evaluator shall perform the following test for each key subject to clearing, including intermediate copies of keys that are persisted encrypted by the TOE:

1. Load the instrumented TOE build in a debugger.

2. Record the value of the key in the TOE subject to clearing.

3. Cause the TOE to perform a normal cryptographic processing with the key from #1.

4. Cause the TOE to clear the key.

5. Cause the TOE to stop the execution but not exit.

6. Cause the TOE to dump the entire memory footprint of the TOE into a binary file.

7. Search the content of the binary file created in #4 for instances of the known key value from #1.

The test succeeds if no copies of the key from #1 are found in step #7 above and fails otherwise.

The evaluator shall perform this test on all keys, including those persisted in encrypted form, to ensure intermediate copies are cleared.

Test 1 - The evaluator used the TOE for normal VPN connections that utilized the associated IPsec cryptographic keys including a client credential. The evaluator logged the IPsec cryptographic keys from the server endpoint before disconnecting the TOE from the VPN connection. The evaluator then dumped the memory on the TOE platform and searched for the IKEv2 keys and client private key, however no traces of the keys were found in memory.

2.1.11 Cryptographic Operation - Hashing - per TD0717 (ASPP14:FCS_COP.1/Hash)

2.1.11.1 ASPP14:FCS_COP.1.1/Hash

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.1 of the ST states that the TOE uses the SHA-256, SHA384, and SHA512 algorithms, and the TOE uses it during IKEv2 authentication message signing and verification and as part of the HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 integrity for IKEv2 and ESP SAs.

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 Section 1.2 of this document.

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

2.1.12.1 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 Section 1.2 of this document.

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

2.1.13.1 ASPP14:FCS_COP.1.1/Sig

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

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

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

ECDSA Algorithm Tests

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

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

RSA Signature Algorithm Tests

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

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

The TOE has been CAVP tested. Refer to the CAVP certificates identified in Section 1.2 of this document.

2.1.14 Cryptographic Operation - Encryption/Decryption - per TD0717 (ASPP14:FCS_COP.1/SKC)

2.1.14.1 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.

Section 3.2 TOE Configuration of the AGD contains information about configuring the TOE with the correct AES encryption/decryption modes and key sizes used for IPsec under CC and CSfC deployments. Additionally, Section 1.5 Excluded Functionality contains a reference to these sections and notes that outside of these configurations, the TOE requires no additional configuration of cryptographic services to be in a CC-compliant mode.

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 Section 1.2 of this document.

2.1.15 Cryptographic Operation (VPNC24:FCS_COP.1/SKC)

2.1.15.1 VPNC24:FCS_COP.1.1/SKC

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: If the TSF implements AES cryptography in support of both credential encryption (per FCS_STO_EXT.1) and IPsec, the evaluator shall examine the TSS to ensure that it clearly identifies the modes and key sizes that are supported for each usage of AES.

The TOE only invokes platform functionality to store credentials to non-volatile memory and does not implement AES cryptography for encryption of credentials (per FCS_STO_EXT.1). Section 6.1 of the ST states that the TOE only supports AES-CBC and GCM as part of IPsec. The TOE relies upon the Android platform for credential (i.e., IKE peer authentication certificates and private keys) encryption/protection.

Component Guidance Assurance Activities: There are no operational beyond what is required by the EA for FCS_COP.1/SKC in the App PP (included below).

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.

Section 3.2 TOE Configuration of the AGD contains information about configuring the TOE with the correct AES encryption/decryption modes and key sizes used for IPsec under CC and CSfC deployments. Additionally, Section 1.5 Excluded Functionality contains a reference to these sections and notes that outside of these configurations, the TOE requires no additional configuration of cryptographic services to be in a CC-compliant mode.

Component Testing Assurance Activities: There are no test EAs beyond what is required by the EA for FCS_COP.1/SKC in the App PP (included below).

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.

The TOE has been CAVP tested. Refer to the CAVP certificates identified in Section 1.2 of this document.

2.1.16 IPsec - per TD0662 (VPNC24:FCS_IPSEC_EXT.1)

2.1.16.1 VPNC24:FCS_IPSEC_EXT.1.1

TSS Assurance Activities: The evaluator shall examine the TSS and determine that it describes how the IPsec capabilities are implemented.

If the TOE is a standalone software application, the evaluator shall ensure that the TSS asserts that all IPsec functionality is implemented by the TSF. The evaluator shall also ensure that the TSS identifies what platform functionality the TSF relies upon to support its IPsec implementation, if any (e.g. does it invoke cryptographic primitive functions from the platform's cryptographic library, enforcement of packet routing decisions by lowlevel network drivers).

If the TOE is part of a general-purpose desktop or mobile OS, the evaluator shall ensure that the TSS describes at a high level the architectural relationship between the VPN client portion of the TOE and the rest of the TOE (e.g. is the VPN client an integrated part of the OS or is it a standalone executable that is bundled into the OS package). If the SPD is implemented by the underlying platform in this case, then the TSS describes how the client interacts with the platform to establish and populate the SPD, including the identification of the platform's interfaces that are used by the client.

In all cases, the evaluator shall also ensure that the TSS describes how the client interacts with the network stack of the platforms on which it can run (e.g., does the client insert itself within the stack via kernel mods, does the client simply invoke APIs to gain access to network services).

The evaluator shall ensure that the TSS describes how the SPD is implemented and the rules for processing both inbound and outbound packets in terms of the IPsec policy. The TSS describes the rules that are available and the resulting actions available after matching a rule. The TSS describes how the available rules and actions form the SPD using terms defined in RFC 4301 such as BYPASS (e.g., no encryption), DISCARD (e.g., drop the packet), and

PROTECT (e.g., encrypt the packet) actions defined in RFC 4301. As noted in section 4.4.1 of RFC 4301, the processing of entries in the SPD is non-trivial and the evaluator shall determine that the description in the TSS is sufficient to determine which rules will be applied given the rule structure implemented by the TOE. For example, if the TOE allows specification of ranges, conditional rules, etc., the evaluator shall determine that the description of rule processing (for both inbound and outbound packets) is sufficient to determine the action that will be applied, especially in the case where two different rules may apply. This description shall cover both the initial packets (that is, no SA is established on the interface or for that particular packet) as well as packets that are part of an established SA.

The TOE is distributed as a standalone software application and is evaluated under the base PP for Application Software

Section 6.1 of the ST states that the TOE is a standalone Android APK and is not integrated into the OS nor is it a standalone executable bundled into the OS package. A user must install the TOE (either through the Playstore or by obtaining the APK file and side-loading it on the Android device). The TOE provides the entirety of both IKEv2 and IPsec/ESP functionality and does not rely upon Android's Linux kernel for any cryptographic processing other than during IKE peer authentication, where the TOE relies upon Android's Keystore to securely store, manage, and utilize user credentials (certificates and private keys). The TOE also relies upon Android's documented, evaluated APIs to enforce packet routing decisions by Android's network drivers). The TOE only provides a "full-tunnel" VPN implementation, which means that the TOE instructs Android (through Android's VPN APIs) to direct all traffic through the PROTECT/encrypt network packet processing.

Guidance Assurance Activities: The evaluator shall examine the operational guidance to verify it describes how the SPD is created and configured. If there is an administrative interface to the client, then the guidance describes how the administrator specifies rules for processing a packet. The description includes all three cases - a rule that ensures packets are encrypted/decrypted, dropped, and allowing a packet to flow in plaintext. The evaluator shall determine that the description in the operational guidance is consistent with the description in the TSS, and that the level of detail in the operational guidance is sufficient to allow the administrator to set up the SPD in an unambiguous fashion. This includes a discussion of how ordering of rules impacts the processing of an IP packet.

If the client is configured by an external application, such as the VPN gateway, then the operational guidance should indicate this and provide a description of how the client is configured by the external application. The description should contain information as to how the SPD is established and set up in an unambiguous fashion. The description should also include what is configurable via the external application, how ordering of entries may be expressed, as well as the impacts that ordering of entries may have on the packet processing.

In either case, the evaluator ensures the description provided In the TSS is consistent with the capabilities and description provided in the operational guidance.

Section 1.6 Security Management of the AGD states that the TOE always enforces a "full-tunnel VPN" and thus subjects all traffic to IPsec/ESP encryption. The TOE does not offer any additional configuration of SPD rules or order other than simply connecting and disconnecting to the configured VPN network. The evaluator found that this description was consistent with the relevant section from the TSS identified above in the TSS Assurance Activity.

The TOE is capable of being configured as stated in Section 3.2 TOE Configuration which details the list of possible configuration options needed, including specifically subsections to ensure this configuration is compliant with CC and CSfC.

Testing Assurance Activities: Depending on the implementation, the evaluator may be required to use a VPN gateway or some form of application to configure the client. For Test 2, the evaluator is required to choose an application that allows for the configuration of the full set of capabilities of the VPN client (in conjunction with the platform). For example, if the client provides a robust interface that allows for specification of wildcards, subnets, etc., it is unacceptable for the evaluator to choose a VPN Gateway that only allows for specifying a single fully qualified IP addresses in the rule.

The evaluator shall perform the following tests:

Test 1: The evaluator shall configure an SPD on the client that is capable of the following: dropping a packet, encrypting a packet, and allowing a packet to flow in plaintext. The selectors used in the construction of the rule shall be different such that the evaluator can generate a packet and send packets to the client with the appropriate fields (fields that are used by the rule - e.g., the IP addresses, TCP/UDP ports) in the packet header. The evaluator performs both positive and negative test cases for each type of rule. The evaluator observes via the audit trail, and packet captures that the TOE exhibited the expected behavior: appropriate packets were dropped, allowed through without modification, was encrypted by the IPsec implementation.

Test 2: The evaluator shall devise several tests that cover a variety of scenarios for packet processing. These scenarios must exercise the range of possibilities for SPD entries and processing modes as outlined in the TSS and operational guidance. Potential areas to cover include rules with overlapping ranges and conflicting entries, inbound and outbound packets, and packets that establish SAs as well as packets that belong to established SAs. The evaluator shall verify, via the audit trail and packet captures, for each scenario that the expected behavior is exhibited, and is consistent with both the TSS and the operational guidance.

Test 1 - Since control over SPDs is limited to connecting and disconnecting the VPN client, the evaluator showed that the platform was implementing BYPASS (allowing all traffic) until the point that the VPN client was connected to a gateway. When the VPN client was connected to the gateway, the evaluator demonstrated that traffic was either PROTECTed when traffic was destined for the VPN and encrypted, or DISCARDed when traffic was not destined for the VPN and ignored.

Test 2 - The testing from the previous test, Test 1, included all various scenarios for packet processing as the control over SPDs is limited to connecting and disconnecting the VPN client. The evaluator tested several combinations of network destinations and protocols which cover the basis of the possibilities of SPD entries.

2.1.16.2 VPNC24:FCS_IPSEC_EXT.1.2

TSS Assurance Activities: The evaluator shall check the TSS to ensure it states that the VPN can be established to operate in tunnel mode, transport mode, or either mode (as selected).

Section 6.1 of the ST states that the TOE provides only tunnel mode IPsec.

Guidance Assurance Activities: The evaluator shall confirm that the operational guidance contains instructions on how to configure the connection in each mode selected.

If both transport mode and tunnel mode are implemented, the evaluator shall review the operational guidance to determine how the use of a given mode is specified.

Section 1.6 Security Management of the AGD states that the TOE always enforces a "full-tunnel VPN" and thus subjects all traffic to IPsec/ESP encryption.

Testing Assurance Activities: The evaluator shall perform the following test(s) based on the selections chosen:

Test 1 [conditional]: If tunnel mode is selected, the evaluator uses the operational guidance to configure the TOE to operate in tunnel mode and also configures a VPN gateway to operate in tunnel mode. The evaluator configures the TOE and the VPN gateway to use any of the allowable cryptographic algorithms, authentication methods, etc. to ensure an allowable SA can be negotiated. The evaluator shall then initiate a connection from the client to connect to the VPN GW peer. The evaluator observes (for example, in the audit trail and the captured packets) that a successful connection was established using the tunnel mode.

Test 2 [conditional]: If transport mode is selected, the evaluator uses the operational guidance to configure the TOE to operate in transport mode and also configures an IPsec peer to accept IPsec connections using transport mode. The evaluator configures the TOE and the endpoint device to use any of the allowed cryptographic algorithms, authentication methods, etc. to ensure an allowable SA can be negotiated. The evaluator then initiates a connection from the TOE to connect to the remote endpoint. The evaluator observes (for example, in the audit trail and the captured packets) that a successful connection was established using the transport mode.

Test 3 [conditional]: If both tunnel mode and transport mode are selected, the evaluator shall perform both Test 1 and Test 2 above, demonstrating that the TOE can be configured to support both modes.

Test 4 [conditional]: If both tunnel mode and transport mode are selected, the evaluator shall modify the testing for FCS_IPSEC_EXT.1 to include the supported mode for SPD PROTECT entries to show that they only apply to traffic that is transmitted or received using the indicated mode.

Test 1 - The TOE only supports tunnel mode. The evaluator configured a VPN gateway to require tunnel mode and attempted to connect the TOE VPN client. The device successfully connected to the VPN gateway and ensured the resulting logs specified tunnel mode was used for the connection.

Tests 2-4 - Not applicable, the TOE does not support transport mode, nor a combination of tunnel and transport mode.

2.1.16.3 VPNC24:FCS_IPSEC_EXT.1.3

TSS Assurance Activities: The evaluator shall examine the TSS to verify that the TSS provides a description of how a packet is processed against the SPD and that if no 'rules' are found to match, that a final rule exists, either implicitly or explicitly, that causes the network packet to be discarded.

Section 6.1 of the ST states that the TOE always enforces a "full-tunnel VPN" and thus subjects all traffic to IPsec/ESP encryption.

Guidance Assurance Activities: The evaluator shall check that the operational guidance provides instructions on how to construct or acquire the SPD and uses the guidance to configure the TOE for the following test.

Section 1.6 Security Management of the AGD states that the TOE always enforces a “full-tunnel VPN” and thus subjects all traffic to IPsec/ESP encryption. The TOE does not offer any additional configuration of SPD rules or order other than simply connecting and disconnecting to the configured VPN network.

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

Test 1: The evaluator shall configure the SPD such that it has entries that contain operations that DISCARD, PROTECT, and (if applicable) BYPASS network packets. The evaluator may use the SPD that was created for verification of FCS_IPSEC_EXT.1.1. The evaluator shall construct a network packet that matches a BYPASS entry and send that packet. The evaluator should observe that the network packet is passed to the proper destination interface with no modification. The evaluator shall then modify a field in the packet header; such that it no longer matches the evaluator-created entries (there may be a 'TOE created' final entry that discards packets that do not match any previous entries). The evaluator sends the packet, and observes that the packet was not permitted to flow to any of the TOE's interfaces.

Test 1 - This was tested as a part of ASPP14:FCS_IPSEC_EXT.1.1 test 1 where the range of possible SPD scenarios was configured and tested. There are no options to configure SPD rules on the TOE of differing specificity or to configure rule ordering outside of enabling/disabling the VPN connection. During this test, the evaluator observed the expected behavior for each of the possible traffic rules.

2.1.16.4 VPNC24:FCS_IPSEC_EXT.1.4

TSS Assurance Activities: The evaluator shall examine the TSS to verify that the algorithms AES-GCM-128 and AES-GCM-256 are implemented. If the ST author has selected either AES-CBC-128 or AES-CBC-256 in the requirement, then the evaluator verifies the TSS describes these as well. In addition, the evaluator ensures that the SHA-based HMAC algorithm conforms to the algorithms specified in the relevant iteration of FCS_COP.1 from the Base-PP that applies to keyed-hash message authentication.

Section 6.1 of the ST states that the TOE provides ESP ciphers of AES-GCM-128, AES-GCM-256, AES-CBC-128, and AES-CBC-256. Additionally, the TOE provides HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA512 for IKEv2 and ESP SA integrity.

Similarly, the evaluator ensured that the HMAC algorithms conformed to the algorithms specified under FCS_COP.1 from ASPP14 as the claims under section 5.1.1.12 of the ST include the 3 claimed functions and their relative key/digest sizes.

Guidance Assurance Activities: The evaluator checks the operational guidance to ensure it provides instructions on how the TOE is configured to use the algorithms selected in this component and whether this is performed through direct configuration, defined during initial installation, or defined by acquiring configuration settings from an environmental component.

Section 3.2 TOE Configuration of the AGD contains information about configuring the TOE with the correct cryptographic algorithms for CC and CSfC deployments.

Testing Assurance Activities: Test 1: The evaluator shall configure the TOE as indicated in the operational guidance configuring the TOE to using each of the AES-GCM-128, and AES-GCM-256 algorithms, and attempt to establish a connection using ESP. If the ST Author has selected either AES-CBC-128 or AES-CBC-256, the TOE is configured to use those algorithms and the evaluator attempts to establish a connection using ESP for those algorithms selected.

Test 1 - The evaluator made an IPsec connection to a VPN gateway using each of the claimed IPsec ciphersuites (AES-GCM-128, AES-GCM-256, AES-CBC-128, and AES-CBC-256).

2.1.16.5 VPNC24:FCS_IPSEC_EXT.1.5

TSS Assurance Activities: The evaluator shall examine the TSS to verify that IKEv1 and/or IKEv2 are implemented. If IKEv1 is implemented, the evaluator shall verify that the TSS indicates whether or not XAUTH is supported, and that aggressive mode is not used for IKEv1 Phase 1 exchanges (i.e. only main mode is used). It may be that these are configurable options.

Section 6.1 of the ST states that the TOE implements only IKEv2 with mandatory support for NAT traversal.

Guidance Assurance Activities: The evaluator shall check the operational guidance to ensure it instructs the administrator how to configure the TOE to use IKEv1 and/or IKEv2 (as selected), and uses the guidance to configure the TOE to perform NAT traversal for the test below. If XAUTH is implemented, the evaluator shall verify that the operational guidance provides instructions on how it is enabled or disabled.

If the TOE supports IKEv1, the evaluator shall verify that the operational guidance either asserts that only main mode is used for Phase 1 exchanges, or provides instructions for disabling aggressive mode.

Section 1.3 TOE Overview of the AGD specifies that the TOE supports IKEv2 connections with IKEv2 VPN gateways. IKEv2 claims mandatory support for NAT traversal and therefore no operational guidance is needed for XAUTH, main mode, or aggressive mode.

Testing Assurance Activities: Test 1:

a. The evaluator shall configure the TOE so that it will perform NAT traversal processing as described in the TSS and RFC 7296, section 2.23. The evaluator shall initiate an IPsec connection and determine that the NAT is successfully traversed.

b. If the TOE supports IKEv1 with or without XAUTH, the evaluator shall verify that this test can be successfully repeated with XAUTH enabled and disabled in the manner specified by the operational guidance. If the TOE only supports IKEv1 with XAUTH, the evaluator shall verify that connections not using XAUTH are unsuccessful. If the TOE only supports IKEv1 without XAUTH, the evaluator shall verify that connections using XAUTH are unsuccessful.

In the case that the VPN gateway enforces the TOE's configuration, the following steps shall be performed to meet the objective of Test 1:

1. Configure the TOE client and VPN gateway to have XAUTH enabled.

2. Attempt the connection and observe that the connection succeeds and that XAUTH is used.

3. Configure the TOE and gateway to have XAUTH disabled.

4. Attempt the connection and observe that the connection succeeds and that XAUTH is not present.

5. Attempt to configure a mismatch between the TOE and gateway (i.e. modify a local configuration setting on the client system)

6. Verify that no IPsec connection is attempted until the gateway corrects the configuration settings

Test 2: [conditional]: If the TOE supports IKEv1, the evaluator shall perform any applicable operational guidance steps to disable the use of aggressive mode and then attempt to establish a connection using an IKEv1 Phase 1 connection in aggressive mode. This attempt should fail. The evaluator shall show that the TOE will reject a VPN gateway from initiating an IKEv1 Phase 1 connection in aggressive mode. The evaluator should then show that main mode exchanges are supported.

In the case that the VPN gateway enforces the TOE's configuration, the following steps should be performed to meet the objective of Test 2:

1. Configure the gateway and TOE client in the appropriate manner per the guidance documentation. (Gateway rejects Aggressive mode, Client rejects aggressive mode)

2. Connect the TOE client to the gateway to obtain the configuration settings.

3. Observe the main mode connection is successful.

4. Disconnect the TOE from the gateway.

5. Attempt to modify the setting for main mode locally on the TOE to force the client to attempt to use aggressive mode.

6. Observe that when the initial connection attempt to the gateway is made, the gateway detects the configuration difference and reapplies the main mode setting before the TOE can attempt an IPsec connection.

7. Configure a peer to have equivalent settings to the VPN gateway (Same ciphers/Authentication/Hash/KEX settings)

8. Tell the TOE that there is a VPN gateway at the location of the peer.

9. Observe that the TOE cannot establish a connection with the peer.

(TD0662 applied)

Test 1 - The evaluator put the device behind a NAT router and initiated a VPN connection between the TOE VPN client and VPN gateway on a secondary network. The packet capture and gateway logs of the traffic demonstrated that the connection through the NAT router was operational. The TOE only supports IKEv2

Test 2 - Not applicable, the TOE does not support IKEv1

2.1.16.6 VPNC24:FCS_IPSEC_EXT.1.6

TSS Assurance Activities: The evaluator shall ensure the TSS identifies the algorithms used for encrypting the IKEv1 and/or IKEv2 payload, and that the algorithms AES-CBC-128, AES-CBC-256 are specified, and if others are chosen in the selection of the requirement, those are included in the TSS discussion.

Section 6.1 of the ST states that the TOE provides IKEv2 ciphers of AES-GMC-128, AES-GMC-256, AES-CBC-128, and AES-CBC-256. The previous TSS entry under VPNC24:FCS_IPSEC_EXT.1.5 states that the TOE only implements IKEv2.

Guidance Assurance Activities: The evaluator checks the operational guidance to ensure it provides instructions on how the TOE is configured to use the algorithms selected in this component and whether this is performed through direct configuration, defined during initial installation, or defined by acquiring configuration settings from an environmental component.

Section 3.2 TOE Configuration of the AGD contains information about configuring the TOE with the correct cryptographic algorithms for CC and CSfC deployments.

Testing Assurance Activities: The evaluator shall use the operational guidance to configure the TOE (or to configure the Operational Environment to have the TOE receive configuration) to perform the following test for each ciphersuite selected:

Test 1: The evaluator shall configure the TOE to use the ciphersuite under test to encrypt the IKEv1 and/or IKEv2 payload and establish a connection with a peer device, which is configured to only accept the payload encrypted using the indicated ciphersuite. The evaluator will confirm the algorithm was that used in the negotiation. The evaluator will confirm that the connection is successful by confirming that data can be passed through the connection once it is established. For example, the evaluator may connect to a webpage on the remote network and verify that it can be reached.

Test 1 - The evaluator attempted an IPsec connection to a VPN gateway using each of the claimed IKE ciphersuites. The evaluator was able to capture each ciphersuite using a packet capture and confirmed the TOE was successfully able to connect using AES-CBC-128, AES-CBC-256, AES-GCM-128, and AES-GCM-256 under the IKEv2 protocol.

2.1.16.7 VPNC24:FCS_IPSEC_EXT.1.7

TSS Assurance Activities: None Defined

Guidance Assurance Activities: The evaluator shall check the operational guidance to ensure it provides instructions on how the TOE configures the values for SA lifetimes. In addition, the evaluator shall check that the guidance has the option for either the Administrator or VPN Gateway to configure Phase 1 SAs if time-based limits are supported. Currently there are no values mandated for the number of packets or number of bytes, the evaluator shall simply check the operational guidance to ensure that this can be configured if selected in the requirement.

Section 1.6 Security Management states that by default, the TOE enforces lifetimes of 24 hours or less for IKEv2 SAs and 7 hours or less for CHILD/ESP SAs. These settings are not configurable. The TOE relies on the VPN Gateway for configuring any stricter values for SA lifetimes.

Testing Assurance Activities: When testing this functionality, the evaluator needs to ensure that both sides are configured appropriately. From the RFC 'A difference between IKEv1 and IKEv2 is that in IKEv1 SA lifetimes were negotiated. In IKEv2, each end of the SA is responsible for enforcing its own lifetime policy on the SA and rekeying the SA when necessary. If the two ends have different lifetime policies, the end with the shorter lifetime will end up always being the one to request the rekeying. If the two ends have the same lifetime policies, it is possible that both will initiate a rekeying at the same time (which will result in redundant SAs). To reduce the probability of this happening, the timing of rekeying requests SHOULD be jittered.'

Each of the following tests shall be performed for each version of IKE selected in the FCS_IPSEC_EXT.1.5 protocol selection:

Test 1 [conditional]: The evaluator shall configure a maximum lifetime in terms of the # of packets (or bytes) allowed following the operational guidance. The evaluator shall establish an SA and determine that once the allowed # of packets (or bytes) through this SA is exceeded, the connection is closed.

Test 2 [conditional]: The evaluator shall construct a test where a Phase 1 SA is established and attempted to be maintained for more than 24 hours before it is renegotiated. The evaluator shall observe that this SA is closed or renegotiated in 24 hours or less. If such an action requires that the TOE be configured in a specific way, the evaluator shall implement tests demonstrating that the configuration capability of the TOE works as documented in the operational guidance.

Test 3 [conditional]: The evaluator shall perform a test similar to Test 2 for Phase 2 SAs, except that the lifetime will be 8 hours or less instead of 24 hours or less.

Test 4 [conditional]: If a fixed limit for IKEv1 SAs is supported, the evaluator shall establish an SA and observe that the connection is closed after the fixed traffic and/or time value is reached.

Test 1 - Not applicable, the TOE does not claim IKEv1 SA lifetimes can be configured based on number of packets/number of bytes.

Test 2 - The evaluator configured a Phase 1 SA lifetime timeout of 25 hours on the VPN gateway. The evaluator then established a connection with the VPN. The IPsec Child SA rekeyed approximately every 7 hours and the IKE SA rekeyed approximately 23 hours after the initial connection.

Test 3 - The evaluator configured a Phase 2 SA lifetime timeout of 9 hours on the VPN gateway. The evaluator then established a connection with the VPN. The IPsec Child SA rekeyed approximately every 7 hours and the IKE SA rekeyed approximately 23 hours after the initial connection.

Test 4 - This test was performed in test 2 where the evaluator tested both Phase 1 and Phase 2 SAs together.

2.1.16.8 VPNC24:FCS_IPSEC_EXT.1.8

TSS Assurance Activities: The evaluator shall check to ensure that the DH groups specified in the requirement are listed as being supported in the TSS. If there is more than one DH group supported, the evaluator checks to ensure the TSS describes how a particular DH group is specified/negotiated with a peer.

Section 6.1 of the ST states that the TOE supports DH groups 19 and 20 (ECP-256 and ECP-384, respectively), and the user can configure the TOE's VPN profiles to use either or both groups for that profiles' VPN connection.

Guidance Assurance Activities: None Defined

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

Test 1: For each supported DH group, the evaluator shall test to ensure that all supported IKE protocols can be successfully completed using that particular DH group.

Test 1 - The evaluator made an IPsec connection to a VPN gateway using each of the claimed DH groups. The evaluator was able to capture a successful connection from the TOE to the gate way with DH groups 19 and 20.

2.1.16.9 VPNC24:FCS_IPSEC_EXT.1.9

TSS Assurance Activities: The evaluator shall check to ensure that, for each DH group supported, the TSS describes the process for generating 'x' (as defined in FCS_IPSEC_EXT.1.9) and each nonce. The evaluator shall verify that the TSS indicates that the random number generated that meets the requirements in this EP is used, and that the length of 'x' and the nonces meet the stipulations in the requirement.

Section 6.1 of the ST states that the TOE supports key exchange groups DH19 and DH20 and generates a secret "x" of size 256 and 384 bits, respectively using the FIPS validated RBG specified in FCS_RBG_EXT.1. When a random number is needed for a nonce, the probability that a specific nonce value will be repeated during the life of a specific IPsec SA is less than 1 in 2^{128} or 2^{192}.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.16.10 VPNC24:FCS_IPSEC_EXT.1.10

TSS Assurance Activities: Assurance Activities for this element are tested through Assurance Activities for FCS_IPSEC_EXT.1.9.

No assurances activities stated for this element outside of those under ASPP14:FCS_IPSEC_EXT.1.9

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.16.11 VPNC24:FCS_IPSEC_EXT.1.11

TSS Assurance Activities: The evaluator shall ensures that the TSS whether peer authentication is performed using RSA, ECDSA, or both.

If any selection with pre-shared keys is chosen in the selection, the evaluator shall check to ensure that the TSS describes how those selections work in conjunction with authentication of IPsec connections.

The evaluator shall ensure that the TSS describes how the TOE compares the peer's presented identifier to the reference identifier. This description shall include whether the certificate presented identifier is compared to the ID payload presented identifier, which fields of the certificate are used as the presented identifier (DN, Common Name, or SAN) and, if multiple fields are supported, the logical order comparison. If the ST author assigned an additional identifier type, the TSS description shall also include a description of that type and the method by which that type is compared to the peer's presented certificate.

Section 6.1 of the ST states that the TOE performs IKE peer authentication using ECDSA only and provides the user the ability to specify the "Server" (i.e., the VPN gateway) identifier. The TOE uses this value to compare against the Distinguished Name (DN) found in the peer's (VPN Gateway's) presented IKE auth certificate.

The TOE does not claim any pre-shared keys under the selection in conjunction with authentication of IPsec connections.

Guidance Assurance Activities: If any selection with 'Pre-shared Keys' is selected, the evaluator shall check that the operational guidance describes any configuration necessary to enable any selected authentication mechanisms.

If any method other than no other method is selected, the evaluator shall check that the operational guidance describes any configuration necessary to enable any selected authentication mechanisms.

The evaluator ensures the operational guidance describes how to set up the TOE to use the cryptographic algorithms RSA, ECDSA, or either, depending which is claimed in the ST.

In order to construct the environment and configure the TOE for the following tests, the evaluator will ensure that the operational guidance also describes how to configure the TOE to connect to a trusted CA, and ensure a valid certificate for that CA is loaded into the TOE as a trusted CA.

The evaluator shall also ensure that the operational guidance includes the configuration of the reference identifiers for the peer.

The TOE does not claim any additional peer authentication methods other than using ECDSA X.509v3 certificates. Section 3.2 TOE Configuration contains details about creating a new VPN profile including importing a certificate through the Android System UI to be used with the TOE.

Similarly, the same section identifies the process of configuring trusted Certificate Authorities through the platform's trusted certificate store. To ensure that only a trusted CA is used, the same stanza states "Assuming the server's certificate matches the configured identifier, the TOE then validates that it can construct a certificate path from the server's certificate through any intermediary CAs to the CA certificate specified by the user in the VPN configuration. If the TOE can successfully build the certificate path, then the TOE will next check the validity of the certificates (e.g., checking its validity dates and that the CA flag is present in the basic constraints section for all CA certs). Assuming the certificates are valid, the TOE finally checks the revocation status of all certificates (starting with the server's certificate and working up the chain). The TOE will reject any certificate for which it cannot determine the validity and reject the connection attempt."

Additionally, section 3.2.2 CC-Specific Configurations states that only ECC client credentials should be configured and the TOE should only be used to connect to servers using ECC server credentials.

To configure reference identifiers for the gateway, the same section states "the TOE provides the user the ability to specify the "Server" (i.e., the VPN gateway) identifier. The TOE uses this value to compare against the Distinguished Name (DN) found in the peer's (VPN Gateway's) presented IKE auth certificate."

Testing Assurance Activities: For efficiency's sake, the testing that is performed here has been combined with the testing for FIA_X509_EXT.2 and FIA_X509_EXT.3 (for IPsec connections and depending on the Base-PP), FCS_IPSEC_EXT.1.12, and FCS_IPSEC_EXT.1.13. The following tests shall be repeated for each peer authentication protocol selected in the FCS_IPSEC_EXT.1.11 selection above:

Test 1: The evaluator shall have the TOE generate a public-private key pair, and submit a CSR (Certificate Signing Request) to a CA (trusted by both the TOE and the peer VPN used to establish a connection) for its signature. The values for the DN (Common Name, Organization, Organizational Unit, and Country) will also be passed in the request. Alternatively, the evaluator may import to the TOE a previously generated private key and corresponding certificate.

Test 2: The evaluator shall configure the TOE to use a private key and associated certificate signed by a trusted CA and shall establish an IPsec connection with the peer.

Test 3: The evaluator shall test that the TOE can properly handle revoked certificates - conditional on whether CRL or OCSP is selected; if both are selected, and then a test is performed for each method. For this current version of the PP-Module, the evaluator has to only test one up in the trust chain (future drafts may require to ensure the validation is done up the entire chain). The evaluator shall ensure that a valid certificate is used, and that the SA is established. The evaluator then attempts the test with a certificate that will be revoked (for each method chosen in the selection) to ensure when the certificate is no longer valid that the TOE will not establish an SA.

Test 4 [conditional]: For each selection made, the evaluator shall verify factors are required, as indicated in the operational guidance, to establish an IPsec connection with the server.

For each supported identifier type (excluding DNs), the evaluator shall repeat the following tests:

Test 5: For each field of the certificate supported for comparison, the evaluator shall configure the peer's reference identifier on the TOE (per the administrative guidance) to match the field in the peer's presented certificate and shall verify that the IKE authentication succeeds.

Test 6: For each field of the certificate support for comparison, the evaluator shall configure the peer's reference identifier on the TOE (per the administrative guidance) to not match the field in the peer's presented certificate and shall verify that the IKE authentication fails.

The following tests are conditional:

Test 7 [conditional]: If, according to the TSS, the TOE supports both Common Name and SAN certificate fields and uses the preferred logic outlined in the Application Note, the tests above with the Common Name field shall be performed using peer certificates with no SAN extension. Additionally, the evaluator shall configure the peer's reference identifier on the TOE to not match the SAN in the peer's presented certificate but to match the Common Name in the peer's presented certificate, and verify that the IKE authentication fails.

Test 8 [conditional]: If the TOE supports DN identifier types, the evaluator shall configure the peer's reference identifier on the TOE (per the administrative guidance) to match the subject DN in the peer's presented certificate and shall verify that the IKE authentication succeeds. To demonstrate a bit-wise comparison of the DN, the evaluator shall change a single bit in the DN (preferably, in an Object Identifier (OID) in the DN) and verify that the IKE authentication fails. To demonstrate a comparison of DN values, the evaluator shall change any one of the four DN values and verify that the IKE authentication fails.

Test 9 [conditional]: If the TOE supports both IPv4 and IPv6 and supports IP address identifier types, the evaluator must repeat test 1 and 2 with both IPv4 address identifiers and IPv6 identifiers. Additionally, the evaluator shall verify that the TOE verifies that the IP header matches the identifiers by setting the presented identifiers and the reference identifier with the same IP address that differs from the actual IP address of the peer in the IP headers and verifying that the IKE authentication fails.

Test 10 [conditional]: If, according to the TSS, the TOE performs comparisons between the peer's ID payload and the peer's certificate, the evaluator shall repeat the following test for each combination of supported identifier types and supported certificate fields (as above). The evaluator shall configure the peer to present a different ID payload than the field in the peer's presented certificate and verify that the TOE fails to authenticate the IKE peer.

Test 1 - This test was performed in VPNC24:FMT_SMF.1/VPN-t1 where a certificate had been configured prior to a successful connection in each case.

Test 2 - This test was performed in VPNC24:FMT_SMF.1/VPN-t1 for IPsec where a certificate (and associated private key) had been configured and used to successfully connect after its certificate chain was validated.

Test 3 - This test was performed in FIA_X509_EXT.1 test case 3 for IPsec where certificate revocation is tested for the peer certificate as well as its issuing chain certificates.

Test 4 - Not applicable - pre-shared keys are not claimed.

Test 5 - This test was performed in FCS_IPSEC_EXT.1.11 test case 6 where IP and FQDN are used for connections where the identifiers match and do not match.

Test 6 - The evaluator iteratively configured a test peer to use an authentication certificate with the correct and incorrect IP address and correct and incorrect DNS address. For the first iteration, the IKE ID matched the certificate IP Address and the connection was established as expected. For the second iteration, the IKE ID did not match the certificate IP Address and the connection was rejected as expected. For the third iteration, the IKE ID matched the certificate FQDN Address and the connection was established as expected. For the fourth iteration, the IKE ID did not match the certificate FQDN Address and the connection was rejected as expected.

Test 7 - Not applicable since only SAN matching is supported.

Test 8 - Not applicable since the TOE does not support DN identifier types.

Test 9 - Not applicable since the TOE supports only IPv4.

Test 10 - Not applicable since the TOE does not support matching the peer identity with the peer certificate.

2.1.16.12 VPNC24:FCS_IPSEC_EXT.1.12

TSS Assurance Activities: Assurance Activities for this element are tested through Assurance Activities for FCS_IPSEC_EXT.1.11.

No assurances activities stated for this element outside of those under ASPP14:FCS_IPSEC_EXT.1.11

Guidance Assurance Activities: Assurance Activities for this element are tested through Assurance Activities for FCS_IPSEC_EXT.1.11.

Assurance Activities for this element are tested through Assurance Activities for FCS_IPSEC_EXT.1.11.

Testing Assurance Activities: None Defined

2.1.16.13 VPNC24:FCS_IPSEC_EXT.1.13

TSS Assurance Activities: Assurance Activities for this element are tested through Assurance Activities for FCS_IPSEC_EXT.1.11.

No assurances activities stated for this element outside of those under ASPP14:FCS_IPSEC_EXT.1.11

Guidance Assurance Activities: Assurance Activities for this element are tested through Assurance Activities for FCS_IPSEC_EXT.1.11.

Assurance Activities for this element are tested through Assurance Activities for FCS_IPSEC_EXT.1.11.

Testing Assurance Activities: None Defined

2.1.16.14 VPNC24:FCS_IPSEC_EXT.1.14

TSS Assurance Activities: The evaluator shall check that the TSS describes the potential strengths (in terms of the number of bits in the symmetric key) of the algorithms that are allowed for the IKE and ESP exchanges. The TSS shall also describe the checks that are done when negotiating IKEv1 Phase 2 and/or IKEv2 CHILD_SA suites to ensure that the strength (in terms of the number of bits of key in the symmetric algorithm) of the negotiated algorithm is less than or equal to that of the IKE SA this is protecting the negotiation.

Section 6.1 of the ST states that the TOE implements RFC 4106 conformant AES-GCM-128 and AES-GCM-256, and RFC 3602 conformant AES-CBC-128 and AES-CBC-256 as encryption algorithms. The TOE implements HMAC-SHA-256, SHA-384, and SHA-512 as authentication algorithms as well as Diffie-Hellman Groups 19 and 20. The encrypted payload for IKEv2 uses AES-CBC-128, AES-CBC-256 as specified in RFC 6379 and AES-GCM-128 and AES-GCM-256 as specified in RFC 5282. The TOE relies upon the VPN Gateway to ensure that the cryptographic algorithms and key sizes negotiated during the IKEv2 negotiation ensure that the security strength of the Phase 1/IKE_SA are greater than or equal to that of the Phase 2/CHILD_SA.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator follows the guidance to configure the TOE to perform the following tests:

Test 1: This test shall be performed for each version of IKE supported. The evaluator shall successfully negotiate an IPsec connection using each of the supported algorithms and hash functions identified in the requirements.

Test 2 [conditional]: This test shall be performed for each version of IKE supported. The evaluator shall attempt to establish an SA for ESP that selects an encryption algorithm with more strength than that being used for the IKE SA (i.e., symmetric algorithm with a key size larger than that being used for the IKE SA). Such attempts should fail.

Test 3: This test shall be performed for each version of IKE supported. The evaluator shall attempt to establish an IKE SA using an algorithm that is not one of the supported algorithms and hash functions identified in the requirements. Such an attempt should fail.

Test 4: This test shall be performed for each version of IKE supported. The evaluator shall attempt to establish an SA for ESP (assumes the proper parameters where used to establish the IKE SA) that selects an encryption algorithm that is not identified in FCS_IPSEC_EXT.1.4. Such an attempt should fail.

Test 1 - All of the algorithms were already tested in FCS_IPSEC_EXT.1.6. The evaluator focused on the hashes for this test and configured the VPN gateway to request each hash (one at a time). The evaluator then successfully connected the device to the VPN gateway with SHA-1, SHA-256, SHA-384, and SHA-512

Test 2 - This property is not enforced by the TOE but rather must be enforced by the VPN gateway since it determines the applicable cipher strengths. As such there is no applicable test.

Test 3 - The evaluator configured the VPN gateway to use an unallowed cipher. The evaluator then attempted to connect the device with the VPN gateway. The connection was refused because the unallowed cipher is not supported.

Test 4 - The evaluator configured the VPN gateway to use the unallowed cipher to establish an SA for ESP in conjunction with the previous test. The evaluator then attempted to connect the device with the VPN gateway. The connection was refused because the unallowed cipher is not supported to establish an SA for ESP.

Component TSS Assurance Activities: In addition to the TSS EAs for the individual FCS_IPSEC_EXT.1 elements below, the evaluator shall perform the following:

If the TOE boundary includes a general-purpose operating system or mobile device, the evaluator shall examine the TSS to ensure that it describes whether the VPN client capability is architecturally integrated with the platform itself or whether it is a separate executable that is bundled with the platform.

The TOE boundary does not include a general-purpose operating system or mobile device, but rather a software application. The TOE is distributed as a separate application as per ASPP14:FPT_TUD_EXT.1.5

Component Guidance Assurance Activities: In addition to the Operational Guidance EAs for the individual FCS_IPSEC_EXT.1 elements below, the evaluator shall perform the following:

If the configuration of the IPsec behavior is from an environmental source, most notably a VPN gateway (e.g through receipt of required connection parameters from a VPN gateway), the evaluator shall ensure that the operational guidance contains any appropriate information for ensuring that this configuration can be properly applied.

Note in this case that the implementation of the IPsec protocol must be enforced entirely within the TOE boundary; i.e. it is not permissible for a software application TOE to be a graphical front-end for IPsec functionality implemented totally or in part by the underlying OS platform. The behavior referenced here is for the possibility that the configuration of the IPsec connection is initiated from outside the TOE, which is permissible so long as the TSF is solely responsible for enforcing the configured behavior. However, it is allowable for the TSF to rely on low-level platform-provided networking functions to implement the SPD from the client (e.g., enforcement of packet routing decisions).

Section 3.2 TOE Configuration of the AGD contains information about configuring the TOE with the correct cryptographic algorithms for CC and CSfC deployments.

Additionally, Section 1.6 Security Management states the TOE includes its own cryptographic library that implements approved cryptographic algorithms that the TOE uses to protect communication between itself and a VPN gateway over an unprotected network using IPsec. The TOE provides the entirety of both IKEv2 and IPsec/ESP functionality and does not rely upon Android's Linux kernel for any cryptographic processing other than during IKE peer authentication, where the TOE relies upon Android's Keystore to securely store, manage, and utilize user credentials (certificates and private keys). The TOE also relies upon Android's documented, evaluated APIs to enforce packet routing decisions by Android's network drivers). In addition, the TOE seeds its DRBG from the Platform.

Component Testing Assurance Activities: As a prerequisite for performing the Test EAs for the individual FCS_IPSEC_EXT.1 elements below, the evaluator shall do the following:

The evaluator shall minimally create a test environment equivalent to the test environment illustrated below. It is expected that the traffic generator is used to construct network packets and will provide the evaluator with the ability manipulate fields in the ICMP, IPv4, IPv6, UDP, and TCP packet headers. The evaluator shall provide justification for any differences in the test environment.

Note that the evaluator shall perform all tests using the VPN client and a representative sample of platforms listed in the ST (for TOEs that claim to support multiple platforms).

The network setup is addressed in the Test Configuration portion of the Detailed Test Report. The TOE VPN client was tested against a common implementation of a VPN Gateway that was modified to produce the manipulated traffic requested for all IPsec testing. Further details about the modifications are provided with individual test cases. Additionally, the evaluator ensured the VPN GW machine was configured to have the ability to capture all traffic on the connected so that all evidence could be sufficiently captured.

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

2.1.17.1 ASPP14:FCS_RBG_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

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

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

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

Section 6.1 of the ST states that the TOE implements DRBG functionality (the AES-256 CTR_DRBG within its OpenSSL library) and seeds it using the ASPP14 proscribed method of calling the Android platform's /dev/random interface.

As a result of the claim, the evaluator ensured that the additional FCS_RBG_EXT.2 elements are included in the ST.

Component Guidance Assurance Activities: None Defined

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

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

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

Platforms: Android....

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

Platforms: Microsoft Windows....

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

Platforms: Apple iOS....

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

Platforms: Linux....

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

Platforms: Oracle Solaris....

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

Platforms: Apple macOS....

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

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

Not applicable, the TOE implements DRBG functionality and instead uses an appropriate platform entropy source to seed its implemented DRBG functionality. Based on the application note for this SFR, the evaluator ensured that the additional FCS_RBG_EXT.2 elements are included in the ST and this report.

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

2.1.18.1 ASPP14:FCS_RBG_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

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

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

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

Implementations Conforming to NIST Special Publication 800-90A

Test 1: The evaluator shall perform 15 trials for the RNG implementation. If the RNG is configurable, the evaluator shall perform 15 trials for each configuration. The evaluator shall also confirm that the operational guidance contains appropriate instructions for configuring the RNG functionality.

If the RNG has prediction resistance enabled, each trial consists of (1) instantiate DRBG, (2) generate the first block of random bits (3) generate a second block of random bits (4) uninstantiate. The evaluator verifies that the second block of random bits is the expected value. The evaluator shall generate eight input values for each trial. The first is a count (0 â€“ 14). The next three are entropy input, nonce, and personalization string for the instantiate operation. The next two are additional input and entropy input for the first call to generate. The final two are additional input and entropy input for the second call to generate. These values are randomly generated. 'generate one block of random bits' means to generate random bits with number of returned bits equal to the Output Block Length (as defined in NIST SP 800-90A).

If the RNG does not have prediction resistance, each trial consists of (1) instantiate DRBG, (2) generate the first block of random bits (3) reseed, (4) generate a second block of random bits (5) uninstantiate. The evaluator verifies that the second block of random bits is the expected value. The evaluator shall generate eight input values for each trial. The first is a count (0 â€“ 14). The next three are entropy input, nonce, and personalization string for the instantiate operation. The fifth value is additional input to the first call to generate. The sixth and seventh are additional input and entropy input to the call to reseed. The final value is additional input to the second generate call.

The following paragraphs contain more information on some of the input values to be generated/selected by the evaluator.

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

Nonce: If a nonce is supported (CTR_DRBG with no Derivation Function does not use a nonce), the nonce bit length is one-half the seed length. Personalization string: The length of the personalization string must be less then or equal to seed length. If the implementation only supports one personalization string length, then the same length can be used for both values. If more than one string length is support, the evaluator shall use personalization strings of two different lengths. If the implementation does not use a personalization string, no value needs to be supplied.

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

The TOE has been CAVP tested. Refer to the CAVP certificates identified in Section 1.2 of this document.

2.1.18.2 ASPP14:FCS_RBG_EXT.2.2

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

The Entropy description is provided in a separate (non-ST and non-Guidance) document that has been approved by NIAP.

Guidance Assurance Activities: None Defined

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

The entropy description is provided in a separate (non-ST and non-Guidance) document that has been delivered to NIAP for approval.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.1.19 Storage of Credentials (ASPP14:FCS_STO_EXT.1)

2.1.19.1 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 state that the TOE's persistent credentials include only the IKE authentication credentials (certificates and their corresponding private keys) for which the TOE relies upon the platform's secure storage using the Android Keystore and KeyChain APIs.

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

The TOE claims to invoke platform-provided functionality to securely store IKEv2 Auth private keys and certificates to non-volatile memory. The evaluator decompiled the TOE application and found several references to Android KeyStore and Android KeyChain APIs. Further testing is done throughout the course of the evaluation (as shown in VPNV24:FMT_SMF.1_VPN where the evaluator walks through this process) that the application does reference certificates that were uploaded into the platform certificate store to be used with the application as a trusted CA or user certificate.

2.2 User data protection (FDP)

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

2.2.1.1 ASPP14:FDP_DAR_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.2 of the ST states that the OE does not store any sensitive data. The application/TOE only provides the capability for the user to create and utilize VPN profiles. The VPN profiles themselves contain no data, only configuration information. Thus, design of the TOE prevents it from storing any data or sensitive data.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: Evaluation activities (after the identification of the sensitive data) are to be performed on all sensitive data listed that are not covered by FCS_STO_EXT.1.

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.

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

Platforms: Android....

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

Platforms: Microsoft Windows....

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

Platforms: Apple iOS....

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

Platforms: Linux....

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

Platforms: Oracle Solaris....

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

Platforms: Apple macOS....

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

Not applicable, the TOE does not claim to store any sensitive data. The evaluation activity therefore does not apply to anything as the evaluator cannot attempt to create sensitive data as there is none.

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

2.2.2.1 ASPP14:FDP_DEC_EXT.1.1

TSS Assurance Activities: None Defined

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

Section 1.6 Security Management of the AGD states the hardware and sensitive-information resources used by the TOE consistent with the claims in the ST by stating "the TOE makes use of network connectivity as it allows users to initiate IPsec VPN connections and accesses no sensitive information repositories."

Testing Assurance Activities: Platforms: Android....

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

Platforms: Microsoft Windows....

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

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

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

Platforms: Apple iOS....

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

Platforms: Linux....

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

Platforms: Oracle Solaris....

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

Platforms: Apple macOS....

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

The TOE application claims to restrict its hardware resources to those required for network connectivity. The evaluator inventoried the AndroidManifest and found several permissions including android.permission.INTERNET and android.permission.ACCESS_NETWORK_STATE. The evaluator examined the remaining set of permissions, however did not find any that could relate to any other hardware resources.

2.2.2.2 ASPP14:FDP_DEC_EXT.1.2

TSS Assurance Activities: None Defined

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

Section 1.6 Security Management of the AGD states the hardware and sensitive-information resources used by the TOE consistent with the claims in the ST by stating "the TOE makes use of network connectivity as it allows users to initiate IPsec VPN connections and accesses no sensitive information repositories."

Testing Assurance Activities: Platforms: Android....

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

Platforms: Microsoft Windows....

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

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

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

Platforms: Apple iOS....

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

Platforms: Linux....

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

Platforms: Oracle Solaris....

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

Platforms: Apple macOS....

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

The TOE application claims to restrict its access to no sensitive data repositories. The evaluator inventoried the AndroidManifest and found several permissions, however did not find any that could relate to sensitive information repositories.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.3 Network Communications (ASPP14:FDP_NET_EXT.1)

2.2.3.1 ASPP14:FDP_NET_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

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

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

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

Platforms: Android....

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

Test 1 - The TOE restricts network communication to user-initiated communication for IPsec VPN connections to a VPN GW. The evaluator sniffed the network to capture traffic while the TOE was used to connect to a VPN GW and demonstrate basic network connectivity. The evaluator then examined the packet capture and found no traffic outside of that which could be attributed to the TOE connecting to the VPN GW, activity outside of the application to demonstrate network connectivity, or traffic that conclusively was from background traffic on the test platform.

Test 2 - The evaluator performed a port scan on the test platform while the TOE was connected to a VPN GW. The evaluator did not find any open or responding ports and concluded that the TOE did not open any ports.

2.2.4 Full Residual Information Protection (VPNC24:FDP_RIP.2)

2.2.4.1 VPNC24:FDP_RIP.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: Requirement met by the platform

The evaluator shall examine the TSS to verify that it describes (for each supported platform) the extent to which the client processes network packets and addresses the FDP_RIP.2 requirement.

Requirement met by the TOE

'Resources' in the context of this requirement are network packets being sent through (as opposed to 'to', as is the case when a security administrator connects to the TOE) the TOE. The concern is that once a network packet is sent, the buffer or memory area used by the packet still contains data from that packet, and that if that buffer is re-used, those data might remain and make their way into a new packet. The evaluator shall check to ensure that the TSS describes packet processing to the extent that they can determine that no data will be reused when processing network packets. The evaluator shall ensure that this description at a minimum describes how the previous data are zeroized/overwritten, and at what point in the buffer processing this occurs.

This requirement is met by the TOE. Section 6.2 of the ST states that the TOE has been designed to ensure that no residual information exists in network packets. When the TOE allocates a new buffer for either an incoming or outgoing network packet, the new packet data will be used to overwrite any previous data in the buffer. If an allocated buffer exceeds the size of the packet, additional space is overwritten (padded) with zeros before the packet is forwarded (to the external network or delivered to the appropriate, internal application).

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.3 Identification and Authentication (FIA)

2.3.1 X.509 Certificate Validation - per TD0669 (ASPP14:FIA_X509_EXT.1)

2.3.1.1 ASPP14:FIA_X509_EXT.1.1

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

Section 6.3 of the ST states that the TOE uses X.509 certificates for IKEv2 authentication. The TOE requires that for each VPN connection, the user specify the client certificate for the TOE to use (the user must have previously loaded such a certificate into the keystore) and specify the CA certificate to which the Gateway/server's certificate must chain. The TOE thus uses the specified certificate when attempting to establish that VPN connection. The TOE validates authentication certificates (including the full path) and checks their revocation status using CRL (compliant with RFC 8603). The TOE processes a VPN connection to a Gateway/server by first comparing the Identification (ID) Payload received from the server against the certificate sent by the server, and if the IP address or FQDN of the certificate does not match the ID, then the TOE does not establish the connection.

Assuming the server's certificate matches the ID, the TOE then validates that it can construct a certificate path from the server's certificate through any intermediary CAs to the CA certificate specified by the user in the VPN configuration. If the TOE can successfully build the certificate path, then the TOE will next check the validity of the certificates (e.g., checking its validity dates and that the CA flag is present in the basic constraints section for all CA certs). Assuming the certificates are valid, the TOE finally checks the revocation status of all certificates (starting with the server's certificate and working up the chain). The TOE will reject any certificate for which it cannot determine the validity and reject the connection attempt.

Guidance Assurance Activities: None Defined

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

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

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

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

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

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

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

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

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

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

The evaluator shall test revocation of the node certificate.

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

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

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

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

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 configured valid server and client certificates. A successful connection was made in each case. The evaluator then configured a server certificate with an invalid certification path by deleting the root CA so that the certificate chain was invalid because of a missing (or deleted) certificate. The connection was refused in each case. The evaluator then configured a test server to send an authentication certificate issued by a Sub CA with no BasicConstraints and with BasicConstraints but the CA Flag set to false and then with an intermediate CA with no keyCertSign purpose and then with an intermediate CA with a path length field set too low. The connection was refused in each case.

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

Test 3 - The TOE only claims support for CRLs. The evaluator configured valid server and client certificates. A successful connection was made in each case. The evaluator then configured a server certificate that was revoked and then a server certificate issued by an intermediate CA that is revoked. The connection was refused in each case.

Test 4 - The evaluator configured valid server and client certificates. A successful connection was made in each case. The evaluator then configured a server certificate issued by an intermediate CA referring to a CRL revocation server where the signer lacks cRLSign, and one issued by an intermediate CA whose issuer CA refers to a CRL revocation server where the signer lacks cRLSign. The connection was refused in each case.

Test 5- The evaluator configured valid server and client certificates. A successful connection was made in each case. The evaluator then configured the server to send an authentication certificate 1) that is valid, 2) that has one byte in the ASN1 field changed, 3) that has one byte in the certificate signature changed, and 4) that has one byte in the certificate public key changed. The connection was refused in each case.

Test 6 - This test was performed with test 5.

Test 7 - This test was performed with test 5.

Test 8 - The evaluator configured valid server and client certificates. A successful connection was made in each case. The evaluator then configured the server to send an authentication certificate with an explicitly defined elliptic curve. The connection was refused in each case.

Test 9 - This was tested with 8.

2.3.1.2 ASPP14:FIA_X509_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

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

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

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

Test 1 - This was tested in conjunction with ASPP14:FIA_X509_EXT.1.1-t1. In that test, the evaluator attempted to connect the TOE to a server using a certificate without a basicConstraints extension and found that the validation of the certificate path fails as part of the validation of the peer certificate belonging to this chain.

Test 2 - This was tested in conjunction with ASPP14:FIA_X509_EXT.1.1-t1. In that test, the evaluator attempted to connect the TOE to a server using a certificate with the CA flag set to false in the basicConstraints extension and found that the validation of the certificate path fails as part of the validation of the peer certificate belonging to this chain.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

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

2.3.2.1 ASPP14:FIA_X509_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.2.2 ASPP14:FIA_X509_EXT.2.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall check the TSS to ensure that it describes how the TOE chooses which certificates to use, and any necessary instructions in the administrative guidance for configuring the operating environment so that the TOE can use the certificates. The evaluator shall examine the TSS to confirm that it describes the behavior of the TOE when a connection cannot be established during the validity check of a certificate used in establishing a trusted channel. The evaluator shall verify that any distinctions between trusted channels are described. If the requirement that the administrator is able to specify the default action, then the evaluator shall ensure that the operational guidance contains instructions on how this configuration action is performed.

Section 6.3 of the ST states that the TOE requires that for each VPN connection, the user specify the client certificate for the TOE to use (the user must have previously loaded such a certificate into the keystore) and specify the CA certificate to which the Gateway/server's certificate must chain. The TOE thus uses the specified certificate when attempting to establish that VPN connection.

Additionally, Section 6.3 of the ST states that the TOE will reject any certificate for which it cannot determine the validity and reject the connection attempt.

According to Section 6.3 of the ST, the TOE only uses X.509 certificates for IKEv2 authentication.

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 & 2 - The evaluator made a valid IPsec connection. The evaluator then defined a non-existent revocation server and attempted the connection a second time and the connection was not accepted as expected. Only CRL is supported for IPsec.

2.3.3 X.509 Certificate Authentication (VPNC24:FIA_X509_EXT.2)

2.3.3.1 VPNC24:FIA_X509_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.3.2 VPNC24:FIA_X509_EXT.2.2

TSS Assurance Activities: This SFR is evaluated in conjunction with FIA_X509_EXT.2 in the App PP

Guidance Assurance Activities: This SFR is evaluated in conjunction with FIA_X509_EXT.2 in the App PP

Testing Assurance Activities: This SFR is evaluated in conjunction with FIA_X509_EXT.2 in the App PP

2.4 Security management (FMT)

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

2.4.1.1 ASPP14:FMT_CFG_EXT.1.1

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

Section 6.4 of the ST states that the TOE requires credentials (i.e., X.509 IKEv2 authentication certificates) for each configured VPN profile; however, the TOE does not install with any such credentials, and the TOE does not manage these credentials, but instead relies upon the Platform (Android Keystore) to handle all user credentials.

Guidance Assurance Activities: None Defined

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

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

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

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

Not applicable. The test cases under FMT_CFG_EXT.1.1 only apply in the application uses any default credentials. The application does not provide any default credentials and furthermore all credentials are managed using the platform certificate storage.

2.4.1.2 ASPP14:FMT_CFG_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

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

Platforms: Android....

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

Platforms: Microsoft Windows....

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

Platforms: Apple iOS....

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

Platforms: Linux....

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

Platforms: Oracle Solaris....

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

Platforms: Apple macOS....

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

The evaluator installed and ran the TOE application before inspecting the filesystem. The evaluator ran the prescribed command in the TOE's data directory which did not return any files. The evaluator did not find any world-writable files in the locations where the application could create data.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

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

2.4.2.1 ASPP14:FMT_MEC_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

The TOE claims to invoke the mechanisms recommended by the platform vendor for storing and setting configuration options.

Section 6.4 of the ST states that the TOE stores user configured or imported VPN profiles in the Android permitted fashion (namely as files within the applications /data/data/package/ directory.

Component Guidance Assurance Activities: None Defined

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

Platforms: Android....

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

Platforms: Microsoft Windows....

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

Platforms: Apple iOS....

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

Platforms: Linux....

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

Platforms: Oracle Solaris....

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

Platforms: Apple macOS....

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

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

The evaluation team queried a TRRT about the behavior of preference storage in the TOE which did not follow above Test Assurance Activity as written and the TRRT agreed that the TOE’s approach was acceptable.

The evaluator installed the TOE and inventoried the /data/data/package file location. The evaluator then ran the application and make security-related changes to its configuration. The evaluator then checked that at least one file at location /data/data/package was updated and examined the file with on board tools to ensure that these changes reflected the security-related changes made previously.

To ensure that this method was platform-supported, the evaluator also browsed online Android documentation and found that SQLite file format used by the vendor was provided along with the Android SDK and sufficient documentation existed on the developer.android.com portal.

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

2.4.3.1 ASPP14:FMT_SMF.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

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

The TOE does not claim any management functions under ASPP14:FMT_SMF.1.1. See VPNC24:FMT_SMF.1/VPN where VPN-specific management functions are fully addressed.

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.

Not applicable, the TOE does not claim any management functions under ASPP14:FMT_SMF.1.1 and therefore there are no management functions to test. See the later SFR, VPNC24:FMT_SMF.1/VPN.1 where further VPN management functions are claimed and tested.

2.4.4 Specification of Management Functions (VPN) (VPNC24:FMT_SMF.1/VPN)

2.4.4.1 VPNC24:FMT_SMF.1.1/VPN

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall check to ensure the TSS describes the client credentials and how they are used by the TOE.

Section 6.4 of the ST states that the TOE provides users the ability to configure, on a per VPN profile basis, the VPN gateway/server, the client credentials/certificate, and the expected reference identified of the VPN gateway/server.

Section 6.3 further elaborates on the use of client credentials by stating that the TOE requires that for each VPN connection, the user specify the client certificate for the TOE to use (the user must have previously loaded such a certificate into the keystore) and specify the CA certificate to which the Gateway/server's certificate must chain. The TOE thus uses the specified certificate when attempting to establish that VPN connection.

Component Guidance Assurance Activities: The evaluator shall check to make sure that every management function mandated in the ST for this requirement is described in the operational guidance and that the description contains the information required to perform the management duties associated with each management function.

The TOE claims the ability to Specify VPN gateways to use for connections, specify client credentials to be used for connections, configure the reference identifier of the peer. Section 3.2 TOE Configuration contains information about all three as it states the following:

The TOE provides the user the ability to specify the “Server” (i.e., the VPN gateway) identifier. The TOE uses this value to compare against the Distinguished Name (DN) found in the peer’s (VPN Gateway’s) presented IKE auth certificate.

The TOE’s UI presents an interface to Androids System UI to import a certificate (chain) and private key in p12/PFX format, or alternatively, the user can separate load the p12 file through Android’s System UI (an MDM Agent or Device Policy Controller can also import p12 certificates as directed by an MDM server). When creating a new VPN profile, the TOE prompts the user to select the certificate/private key they wish the TOE to use during IKE authentication. The TOE does not install with any default credentials, nor does it store any credentials imported by the user as this is maintained by the platform (Android Keystore) and is simply leveraged by the TOE.

Component Testing Assurance Activities: The evaluator shall test the TOE's ability to provide the management functions by configuring the TOE according to the operational guidance and testing each management activity listed in the ST.

The evaluator shall ensure that all management functions claimed in the ST can be performed by completing activities described in the AGD. Note that this may be performed in the course of completing other testing.

The TOE claims the following VPN management functions: specify VPN gateways to use for connections, specify client credentials to be used for connections, and configure the reference identifier of the peer. The evaluator demonstrated the TOE’s ability to do all three at the same time by configuring a new VPN configuration with a specified gateway, server identifier, and client credentials and ensuring that this could be used to connect to a VPN gateway that required correct configuration of all three. Since the AGD makes references to imported configurations, the evaluator demonstrated the TOE’s ability to additionally import and edit an analogous configuration file and showed that this had the same behavior.

2.5 Privacy (FPR)

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

2.5.1.1 ASPP14:FPR_ANO_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.5 of the ST states that the TOE does not transmit any PII over a network.

Component Guidance Assurance Activities: None Defined

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

Not applicable, the TOE does not collect or transmit any PII over a network.

2.6 Protection of the TSF (FPT)

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

2.6.1.1 ASPP14:FPT_AEX_EXT.1.1

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

Section 6.6 of the TSS states that the TOE's native libraries (built using Android's NDK) enable ASLR and stack protection by fPIC, -DOPENSSL_PIC, and the -fstack-protector-all flags. Furthermore, the application does not allocate any memory region with execute permissions (and thus prevents allocation of any memory region with both write and execute permissions).

Guidance Assurance Activities: None Defined

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

Platforms: Android....

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

Platforms: Microsoft Windows....

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

Platforms: Apple iOS....

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

Platforms: Linux....

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

Platforms: Oracle Solaris....

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

Platforms: Apple macOS....

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

The evaluator performed a dynamic analysis to determine that no memory mappings are placed at an explicit and consistent address following the test requirements for Android. For this test, the evaluator used several devices to collect memory mappings for the application, however compared the results from a singular device with itself at a later time. After collecting the first instance of mappings, the evaluator uninstalled the application, rebooted the device, and reinstalled the application to collect the second instance of mappings. The evaluator compared the two outputs and found that they were sufficiently randomized.

2.6.1.2 ASPP14:FPT_AEX_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

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

Platforms: Android....

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

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

o mprotect is never invoked.

Platforms: Microsoft Windows....

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

Platforms: Apple iOS....

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

Platforms: Linux....

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

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

o mprotect is never invoked with the PROT_EXEC permission.

Platforms: Oracle Solaris....

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

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

o mprotect is never invoked with the PROT_EXEC permission.

Platforms: Apple macOS....

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

The evaluator decompiled the TOE application and performed a static analysis on the resulting files. The evaluator searched for uses of mmap and mprotect. The evaluator did not find any instances of the keyword mprotect. The evaluator did find several instances of the keyword mmap but determined the majority of these to conclusively be references to keywords containing the word mmap (such as newSetFromMap) or references to a similarly named Java mmap function which does not use these same protections. The evaluator examined any references to the keyword mmap that could not be conclusively ruled out and found no references to parameters passed in including PROT_WRITE, PROT_EXEC, or their enumerated values.

2.6.1.3 ASPP14:FPT_AEX_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

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

Platforms: Android....

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

Platforms: Microsoft Windows....

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

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

Platforms: Apple iOS....

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

Platforms: Linux....

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

Platforms: Oracle Solaris....

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

Platforms: Apple macOS....

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

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

2.6.1.4 ASPP14:FPT_AEX_EXT.1.4

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

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

Platforms: Android....

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

Platforms: Microsoft Windows....

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

Platforms: Apple iOS....

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

Platforms: Linux....

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

Platforms: Oracle Solaris....

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

Platforms: Apple macOS....

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

The evaluator ran the application and determined where it could write files. The evaluator determined that the TOE could write files to its internal data directory. The evaluator searched the data directory and found no executables located under that location.

2.6.1.5 ASPP14:FPT_AEX_EXT.1.5

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

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

Platforms: Microsoft Windows....

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

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

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

xor rcx, (...)

call (...)

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

Tools such as Canary Detector may help automate these activities.

The evaluator decompiled the TOE executable and made note of every native executable included with the TOE software. The evaluator found several ELF executable files that could be further analyzed with binary search utilities and searched each for the __stack_chk_fail symbol. The evaluator found that the symbol was present in each of the bundled native executables signifying that stack-based buffer overflow protection was present.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

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

2.6.2.1 ASPP14:FPT_API_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.6 of the TSS contains a list of APIs used by the application that all are a part of the Android Platform API.

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.

Test 1 - The evaluator obtained a list of all APIs used by the TOE. The evaluator browsed the Android Developer's website at https://developer.android.com/reference/. The evaluator found references to each of the above classes listed as documented APIs under the Android Platform.

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

2.6.3.1 ASPP14:FPT_IDV_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

The TOE claims to be versioned with an APK version number, therefore section 6.6 of the ST expands on this by stating that the TOE uses a major, minor, and build number.

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 TOE application is versioned with an APK version number, which does not comply with ISO/IEC 19770-2:2015 SWIG tags. As a result, the evaluator references ASPP14:FPT_TUD_EXT.1.1-t1 where the TOE was installed and the evaluator checked for the existence of version information.

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

2.6.4.1 ASPP14:FPT_LIB_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

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

The evaluator installed the TOE application and surveyed the installation directory for dynamic libraries. The evaluator found that this only contained several vendor-developed libraries which was consistent with the claims in the ST.

2.6.5 TSF Self-Test (VPNC24:FPT_TST_EXT.1/VPN)

2.6.5.1 VPNC24:FPT_TST_EXT.1.1/VPN

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.6.5.2 VPNC24:FPT_TST_EXT.1.2/VPN

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: Except for where it is explicitly noted, the evaluator is expected to check the following information regardless of whether the functionality is implemented by the TOE or by the TOE platform.

The evaluator shall examine the TSS to ensure that it details the self-tests that are run by the TSF on start-up; this description should include an outline of what the tests are actually doing (e.g., rather than saying 'memory is tested', a description similar to 'memory is tested by writing a value to each memory location and reading it back to ensure it is identical to what was written' shall be used). The evaluator shall ensure that the TSS makes an argument that the tests are sufficient to demonstrate that the TSF is operating correctly. If some of the tests are performed by the TOE platform, the evaluator shall check the TSS to ensure that those tests are identified, and that the ST for each platform contains a description of those tests. Note that the tests that are required by this component are those that support security functionality in the VPN Client PP-Module, which may not correspond to the set of all self-tests contained in the platform STs.

The evaluator shall examine the TSS to ensure that it describes how the integrity of stored TSF executable code is cryptographically verified when it is loaded for execution. The evaluator shall ensure that the TSS makes an argument that the tests are sufficient to demonstrate that the integrity of stored TSF executable code has not been compromised. The evaluator shall check to ensure that the cryptographic requirements listed are consistent with the description of the integrity verification process.

The evaluator also ensures that the TSS (or the operational guidance) describes the actions that take place for successful (e.g. hash verified) and unsuccessful (e.g., hash not verified) cases. For checks implemented entirely by the platform, the evaluator ensures that the operational guidance for the TOE references or includes the platform-specific guidance for each platform listed in the ST.

Section 6.6 of the ST contains a description of the start-up cryptographic self-tests including an outline of the various cryptographic functions that are tested against known answers. The evaluator ensured that these sufficiently covered the algorithms claimed under this evaluation. The TOE only claims to perform its own cryptographic self-tests.

The TOE claims the TOE platform shall provide the capability to verify the integrity of the store TSF executable code through the use of digital signature verification. Section 6.6 of the ST confirms this by stating that the TOE relies upon the Platform for storage, integrity, and verification of the TOE's executable code.

Component Guidance Assurance Activities: Except for where it is explicitly noted, the evaluator is expected to check the following information regardless of whether the functionality is implemented by the TOE or by the TOE platform.

If not present in the TSS, the evaluator ensures that the operational guidance describes the actions that take place for successful (e.g. hash verified) and unsuccessful (e.g., hash not verified) cases. For checks implemented entirely by the platform, the evaluator ensures that the operational guidance for the TOE references or includes the platform-specific guidance for each platform listed in the ST.

Section 3.6 Self-Test of the AGD states that the TOE performs a series of self-tests upon loading/execution. These include cryptographic algorithm self-tests for all of the claimed algorithms within its OpenSSL library. Upon failure of any of the individual self-tests, the TOE will halt execution and display an error message to the user before exiting the application. Only after correctly passing all self-tests will the TOE permit any security functions. Details about self-test results can be found under the device internal logcat storage.

Component Testing Assurance Activities: Except for where it is explicitly noted, the evaluator is expected to check the following information regardless of whether the functionality is implemented by the TOE or by the TOE platform.

The evaluator shall perform the following tests:

Test 1: The evaluator performs the integrity check on a known good TSF executable and verifies that the check is successful.

Test 2: The evaluator modifies the TSF executable, performs the integrity check on the modified TSF executable and verifies that the check fails.

Test 1 - The evaluator started up the TOE which automatically triggers the TOE's integrity checks. Upon passing the integrity tests, the TOE started up and was operational

Test 2 - The evaluator installed and started up a modified version of the TOE which was compiled to automatically fail its start-up integrity checks. Upon failing the integrity checks, the TOE reported these errors in the test platform's internal logcat storage, displayed an error to the user, and then shut down the TOE before it could be used for any operations.

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

2.6.6.1 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 3.7 Current Version and Trusted Updates of the AGD states DataSoft makes TOE updates through the APK package format and distributes updated APKs through the Google Play Store. To initiate an update, the user can use the Play Store application to download any available update. An update is determined successful if the version number is updated and the Play Store no longer reports an update is available. DataSoft signs the Secure Tactical VPN Client APK with a unique developer private key to ensure authenticity of updates which is then verified by the Google Play Store and EUD.

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.

Test 1 - The evaluator used the Google Playstore to check for any available updates. The platform Playstore application did not report back any updates were available.

2.6.6.2 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 3.7 Current Version and Trusted Updates of the AGD states the current version of the TOE is reported alongside application information under the EUD's Settings application. This information can be accessed by long pressing the TOE icon on the EUD and then selecting App Info and scrolling to the bottom. The TOE uses a major, minor, and build number.

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.

Test 1 - The evaluator queried the version for the application under ASPP14:FPT_TUD_EXT.1.1-t1 using the platform Settings application. The resulting version was found to match the documented and installed version

2.6.6.3 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.

Test 1 - The evaluator installed the TOE and used the test platform's shell to inventory the directory where the TOE executables are saved. The evaluator then saved a copy of a timestamp and md5 hash of each file in the directory. The evaluator then used the TOE to encrypt a file, decrypt a file, change the password, and restarted the TOE service. Afterward, the evaluator returned to the test platform's shell and obtained a new timestamp and MD5 hash for each executable in the installation directory. The evaluator verified that these were identical.

2.6.6.4 ASPP14:FPT_TUD_EXT.1.4

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

Section 6.6 of the ST states that DataSoft makes VPN Client updates through the APK package format and distributes updated APKs both directly to customers as well as through the Google Playstore.   Additionally, the same section states DataSoft signs their VPN Client APK with their unique developer private key to ensure authenticity of updates. That DataSoft unique developer private key corresponds to DataSoft's developer public key registered with Google's PlayStore.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.6.6.5 ASPP14:FPT_TUD_EXT.1.5

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

Section 6.6 of the ST states that DataSoft makes VPN Client updates through the APK package format and distributes updated APKs both directly to customers as well as through the Google Playstore.

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.

Test 1 - The TOE claims to be distributed as an additional software package to the platform OS. As a result, the 'with the platform' test does not apply and the evaluator selected and performed the tests under FPT_TUD_EXT.2

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

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

2.6.7.1 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.

Test 1 - The TOE claims to be distributed using the format (APK) of the platform (Android)-supported package manager. The vendor provided the application in the form of an APK file for Android and this was used for the entirety of testing.

2.6.7.2 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)

Test 1 - This requirement is inherently met as the platform forces applications to write data within the application working directory.

2.6.7.3 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 6.6 of the ST states that DataSoft makes VPN Client updates through the APK package format and distributes updated APKs both directly to customers as well as through the Google Playstore.   Additionally, the same section states DataSoft signs their VPN Client APK with their unique developer private key to ensure authenticity of updates. That DataSoft unique developer private key corresponds to DataSoft's developer public key registered with Google's PlayStore.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.7 Trusted path/channels (FTP)

2.7.1 Protection of Data in Transit - per TD0743 – per TD0743 (ASPP14:FTP_DIT_EXT.1)

2.7.1.1 ASPP14:FTP_DIT_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

The TOE does not invoke any platform-provided functionality to encrypt transmitted data.

Component Guidance Assurance Activities: None Defined

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

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

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

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

Platforms: Android....

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

Platforms: Apple iOS....

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

Test 1 - The evaluator examined the packet capture from VPNC24:FCS_IPSEC_EXT.1-t1 as it contained network traffic of the TOE while it was being exercised for its general operations. The TOE claims to encrypt its transmitted data with IPsec as defined in the PP-Module for VPN Client. The evaluator verified that the packet capture was encrypted with IPsec.

Test 2 - the evaluator reexamined the packet capture from VPNC24:FCS_IPSEC_EXT.1-t1. Although there is no sensitive data identified by the application, the evaluator examined the traffic and found that once the IPsec handshake completed, all platform traffic was being encrypted inside ESP packets and not being sent in plaintext.

Test 3 - The TOE does not contain any credentials of its own, however it does transmit platform client credentials as a part of the IPsec handshake. During the communication, the evaluator used a credential with a fixed private key that was only transmitted to the VPN server as a part of the IPsec handshake. Since this should not appear in plaintext anywhere in the packet capture, the evaluator searched for the plaintext certificate and private key in the packet capture for the corresponding client credential used and could not find any evidence of it present.

2.7.2 Protection of Data in Transit (VPNC24:FTP_DIT_EXT.1)

2.7.2.1 VPNC24: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 IPsec, refer to the EA for FCS_IPSEC_EXT.1. If other protocols are selected for FTP_DIT_EXT.1, refer to the EA for FTP_DIT_EXT.1 in the App PP (included below).

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

VPNC24: For IPsec, refer to the EA for FCS_IPSEC_EXT.1. If other protocols are selected for FTP_DIT_EXT.1, refer to the EA for FTP_DIT_EXT.1 in the App PP (included below).

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

For IPsec, refer to the EA for FCS_IPSEC_EXT.1. The TOE does not claim any additional protocols for encrypting transmitted data. The TOE does not invoke any platform-provided functionality for encrypting transmitted data.

Component Guidance Assurance Activities: For IPsec, refer to the EA for FCS_IPSEC_EXT.1.

VPNC24: For IPsec, refer to the EA for FCS_IPSEC_EXT.1.

Refer to the EA for FCS_IPSEC_EXT.1

Component Testing Assurance Activities: For IPsec, refer to the EA for FCS_IPSEC_EXT.1. If other protocols are selected for FTP_DIT_EXT.1, refer to the EA for FTP_DIT_EXT.1 in the App PP (included below).

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.

VPNC24: For IPsec, refer to the EA for FCS_IPSEC_EXT.1. If other protocols are selected for FTP_DIT_EXT.1, refer to the EA for FTP_DIT_EXT.1 in the App PP (included below).

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.

The TOE claims all sensitive data is encrypted with IPsec and no other protocols. For IPsec, refer to the EA for FCS_IPSEC_EXT.1.

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.

There are no specific assurance activities associated with these SARs, except ensuring the information is provided. The requirements on the content are implicitly assessed by the virtue of other assurance activities performed.

3.2 Guidance documents (AGD)

3.2.1 Operational User Guidance (AGD_OPE.1)

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

Section 6.1 Security Management of the AGD states the TOE includes its own cryptographic library that implements approved cryptographic algorithms that the TOE uses to protect communication between itself and a

VPN gateway over an unprotected network using IPsec. Additionally, it states The TOE contains several other configurable security settings as addressed under Section 3.2 TOE Configuration. See this section for additional information on these configuration options. Beyond configuring CC-specific or CSfC-specific configuration options under individual VPN profile, no additional configuration of the cryptographic engine is needed.

Section 3.7 Current Version and Trusted Updates of the AGD states DataSoft makes TOE updates through the APK package format and distributes updated APKs through the Google Play Store. To initiate an update, the user can use the Play Store application to download any available update. An update is determined successful if the version number is updated and the Play Store no longer reports an update is available. DataSoft signs the Secure Tactical VPN Client APK with a unique developer private key to ensure authenticity of updates which is then verified by the Google Play Store and EUD.

Section 1.4 Operational Environment details components of the operational environment that are outside the TOE include a certificate authority, mobile platform, and VPN gateway (with specific references to state that the evaluation does not cover any aspects of the separately CC-evaluated DataSoft RAP-117 VPN Gateway). Additionally, Section 1.5 Excluded Functionality contains a reference to sections for configuring the TOE in a CC or CSfC-complaint mode and operating outside of these configurations are not covered by this evaluation. Lastly, this section also identifies SSL Tunnel with DLTS tunneling options is also not a evaluated behavior.

3.2.2 Preparative Procedures (AGD_PRE.1)

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

Section 1.4 Operational Environment details the components of the operational environment to support the TOE. The evaluator ensured that the mobile platforms described here included any Android mobile device platform running Android 11, 12, or 13 using a kernel earlier than version 5.6 which is consistent with the claims made in the ST.

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.

Section 1.3 TOE Overview of the ST states that the Target of Evaluation (TOE) is the DataSoft Secure Tactical VPN Client for Android (SW version 2.3.7). The evaluator found that this was consistent with section 1.0 Introduction of the AGD that states the document is for the administration of the DataSoft Secure Tactical VPN Client for Android, version 2.3.7 and consistent with the TOE samples provided for testing. The evaluator examined the vendor's website and only Playstore listings to ensure that this information was sufficient enough to distinguish the TOE from other products.

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.

See section 3.3.1 above for an explanation of how all CM items are identified.

Section 1.4 Operational Environment of the AGD identifies the Mobile Platform supported for the TOE are any Android mobile device platforms running Android 11, 12, or 13 using a kernel earlier than version 5.6. Section 1.6 Security Management of the AGD also states that the TOE is compiled with all necessary compilation flags to ensure that bugger overflow protection mechanisms are invoked and no additional platform mechanisms need to be specifically enabled.

3.3.3 Timely Security Updates (ALC_TSU_EXT.1)

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

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

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

Section 6.6 of the TSS states the vendor provides timely security updates for the TOE in case vulnerabilities have been discovered. Reported vulnerabilities and defects are investigated and rated based on the threat and result of the impact analysis and then scheduled for an upcoming bug fix release based on the severity. The vendor aims for security updates as soon as possible with a maximum of 30 days. Third party library updates (OpenSSL) are also included as a part of the TOE's update. The vendor actively monitors both internal and third-party components and accepts vulnerability reports through the DataSoft email support address (support@DataSoft.com).

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.

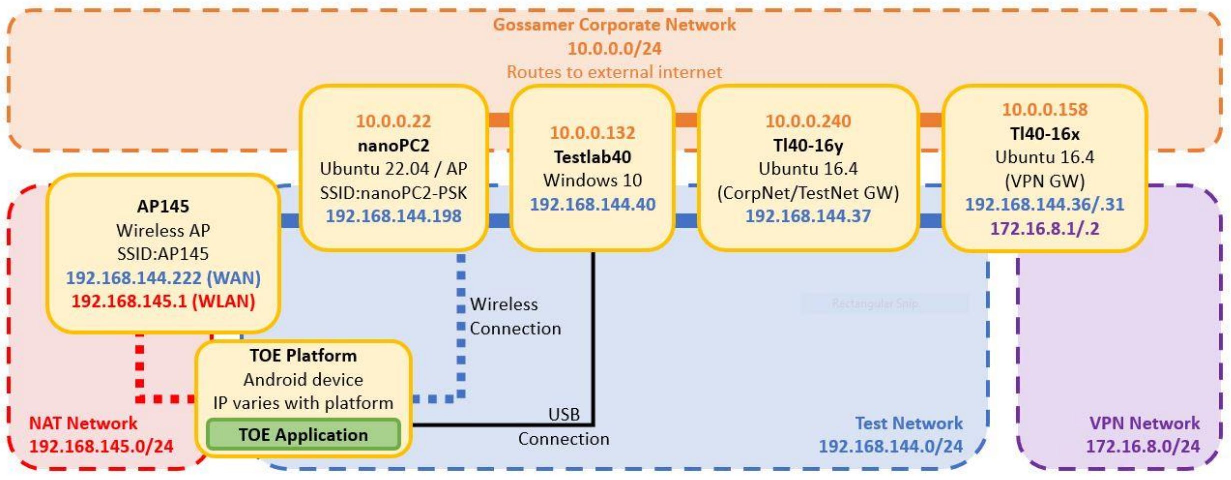
While it is not necessary to have one test case per test listed in an Assurance Activity, the evaluator must document in the test plan that each applicable testing requirement in the ST is covered. The test plan identifies the platforms to be tested, and for those platforms not included in the test plan 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 TOE was made available at the Gossamer testing laboratory. When performing testing, the evaluator configured the TOE into CC mode as described in the AGD. The following diagrams show the evaluator test configurations:



TOE Platforms:

- DataSoft Secure Tactical VPN Client for Android, Version 2.3.7

Supporting Products:

- Test Platforms – used to install and run the TOE application as well as capture and manage test evidence, such as screen captures and logcats

    - Samsung S20 Tactical Edition (TE) running Android 11
    - Google Pixel 5 running Android 11
    - Google Pixel 4a-5G running Android 12
    - Google Pixel 5a-5G running Android 13
- Test Machine -
    - Windows 10
- Test Machine 2
    - Ubuntu 22.04
- VPN Gateway Machine
    - Ubuntu 16.04
- Test Network Gateway machine 2
    - Ubuntu 16.04
- AP145
    - UTT AC650W Access Point

Supporting Software:

- Standard Ubuntu utilities – grep, find, file, head, unzip, tr, strings, nm, readelf, objdump, openssl, ping, curl, ls, cd, ps
- Strongswan VPN Server version 5.3.5
- Tcpdump version 4.9.3
- apktool version 2.6.1
- Nmap version 7.92
- Putty version 0.74
- Wireshark version 4.0.3
- Notepad++ version 8.4.8
- Android Debug Bridge (adb) version 33.0.2-8557947
- Gossamer tools

3.5 Vulnerability assessment (AVA)

3.5.1 Vulnerability Survey (AVA_VAN.1)

Assurance Activities: The evaluator shall generate a report to document their findings with respect to this requirement. This report could physically be part of the overall test report mentioned in ATE_IND, or a separate document. The evaluator performs a search of public information to find vulnerabilities that have been found in similar applications with a particular focus on network protocols the application uses and document formats it parses. The evaluator shall also run a virus scanner with the most current virus definitions against the application files and verify that no files are flagged as malicious. The evaluator documents the sources consulted and the vulnerabilities found in the report. For each vulnerability found, the evaluator either provides a rationale with respect to its non-applicability, or the evaluator formulates a test (using the guidelines provided in ATE_IND) to confirm the vulnerability, if suitable. Suitability is determined by assessing the attack vector needed to take advantage of the vulnerability. If exploiting the vulnerability requires expert skills and an electron microscope, for instance, then a test would not be suitable and an appropriate justification would be formulated.

The evaluator searched the National Vulnerability Database (https://web.nvd.nist.gov/vuln/search), Vulnerability Notes Database (http://www.kb.cert.org/vuls/) on 7/18/2023 with the following search terms: "DataSoft Tactical Secure Mobility VPN Client", "DataSoft Secure Tactical Mobility VPN Client", "DataSoft Secure Tactical VPN Client", "Secure Tactical VPN Client", "DataSoft VPN Client", "DataSoft Corporation", "DataSoft", "OpenSSL", "Strongswan".

The search did not discover any known vulnerabilities in the TOE.

The Test Activity to run a virus scanner is only applicable for Windows, Linux, macOS and Solaris. Since the TOE is an android application, this test activity is not applicable