www.GossamerSec.com

Assurance Activity Report for Shift5, Inc. Software File-Based Encryption

Version 0.2
07/23/26

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

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

Document: AAR-VID11663

Revision History

Revision	Date	Authors	Summary
Version 0.1	07/10/26	Gossamer	Initial draft
Version 0.2	07/22/26	Gossamer	First Round Comments

The TOE Evaluation was Sponsored by:

Shift5 Inc.
1100 Wilson Blvd, Ste 2100
Rosslyn, VA 22209

Evaluation Personnel:

- Reed Eberly

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 Shift5, Inc. Software File-Based Encryption ASPP14/FE10/PKGTLS11 evaluation. This document contains a description of the assurance activities and associated results as performed by the evaluators.

1.1 Test Equivalence

The evaluator fully tested the TOE containerized application on a Shift5 Manifold hardware running the SUSE Linux Micro 6.0-based platform image and an internal Rancher Government Solutions Rancher Kubernetes Engine 2 (RKE2) running on an Atom C3708 CPU. The TOE is primarily a containerized application that provides its own userspace runtime which abstracts away many of the dependencies from the base image. For the purposes of this evaluation, the TOE is compatible with any of the SUSE Linux Micro 6-based images Shift5 provides for their product line that ship with the TOE pre-installed.

1.2 CAVP Algorithms

The TOE provides two cryptographic libraries to perform the necessary cryptographic functions in accordance with the following NIST standards and has received the following CAVP algorithm certificates.

The TOE uses its Shift5 Libgcrypt Cryptography library version 1.11.2 to support its FDE functionality.

SFR	Algorithm	NIST Standard	Cert#
ASPP14:FCS_COP.1/Hash	SHA-256/384 Hashing	FIPS 180-4	A7834
ASPP14:FCS_COP.1/KeyedHash	HMAC-SHA-384	FIPS 198-1 & 180-4	A7834
ASPP14:FCS_COP.1/SKC	AES-256 XTS Encrypt/Decrypt	NIST SP 800-38E	A7834
FE10:FCS_COP.1(7) (Key Encryption)	AES-256 CBC Encrypt/ Decrypt	NIST SP 800-38A	A7834
ASPP14:FCS_RBG_EXT.2	HMAC_DRBG (SHA256)	SP 800-90A	A7834

Table 1 Shift5 Libgcrypt-based Cryptographic Library Cryptographic Algorithms

The TOE uses its Shift5 OpenSSL 3.X FIPS Provider Library version 3.1.2to support password validation and all of its TLS functionality.

SFR	Algorithm	NIST Standard	Cert#
ASPP14:FCS_CKM.1/AK (Key Gen)	ECDSA ECC key gen P-256, P-384, P-521	FIPS 186-5, ECDSA	A7329
ASPP14:FCS_CKM.2 (Key Establishment)	ECC-based key exchange	SP 800-56A, KAS ECC	A7329
ASPP14:FCS_COP.1/SKC	AES-256 CBC Encrypt/ Decrypt	NIST SP 800-38A	A7329

SFR	Algorithm	NIST Standard	Cert#
	AES-128/256 GCM Encrypt / Decrypt	NIST SP 800-38D	
ASPP14:FCS_COP.1/Hash	SHA-256/384 Hashing	FIPS 180-4	A7329
ASPP14:FCS_COP.1/KeyedHash	HMAC-SHA-384	FIPS 198-1 & 180-4	A7329
ASPP14:FCS_COP.1/Sig	ECDSA Sign/Verify P-256, 384, 521	FIPS 186-5, ECDSA	A7329
ASPP14:FCS_RBG_EXT.2	AES-256 CTR_DRBG	SP 800-90A	A7329

Table 2 Shift5 OpenSSL-based Cryptographic Library Cryptographic Algorithms

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:

- Shift5, Inc. Software File-Based Encryption Security Target, Version 0.4 07/23/2026 [ST]
- Detailed Test Report for Shift5, Inc. Software File-Based Encryption, Version 0.2, 07/23/2026 [DTR]
- Shift5 File – Based Encryption (FBE) Administrator Guidance Document (AGD), Version 1.3, April 2026 [AGD]

The vendor did not provide a separate Key Management Document (KMD), and instead opted to include the relevant security information directly in the Security Target. All assurance activities for the KMD have been verified against the ST.

2.1 Cryptographic support (FCS)

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

2.1.1.1 ASPP14:FCS_CKM.1.1/AK

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

Section 6.1 of the [ST] states the TOE generates P-256, P-384, and P-521 ECDHE keys as a part of TLS connections from its Administrator WebUI used to start/stop the encryption service. Additionally, the TOE webserver creates a new web server P-521 ECDSA TLS certificate and private key during provisioning. The TOE's asymmetric key

generation is implemented and restricted to the TOE's OpenSSL library which is responsible for cryptography in its WebUI webserver.

The application implements functionality to generate asymmetric cryptographic 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 generation scheme(s) and key size(s) for all uses defined in this PP.

Section 1.3 of AGD states The TOE implements CAVP-certified cryptographic algorithms which are verified as a part of this evaluation. No configuration is needed for the cryptographic modules in order to be compliant. The use of any other cryptographic components beyond those that are identified are neither evaluated nor tested during the TOE's Common Criteria 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 or FIPS PUB 186-5 RSA Schemes

The evaluator shall verify the implementation of RSA Key Generation by the TOE using the Key Generation test. This test verifies the ability of the TSF to correctly produce values for the key components including the public verification exponent e, the private prime factors p and q, the public modulus n and the calculation of the private signature exponent d. Key Pair generation specifies 5 ways (or methods) to generate the primes p and q. These include:

1. Random Primes:

Provable primes

Probable primes

2. Primes with Conditions:

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

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

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

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

n = p*q,

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

GCD(p-1,e) = 1,

GCD(q-1,e) = 1,

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

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

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

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

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

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

Key Generation for Elliptic Curve Cryptography (ECC)

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

FIPS 186-4 or FIPS 186-5 Public Key Verification (PKV) Test For each supported NIST curve, i.e., P256, P-384 and P-521, the evaluator shall generate 10 private/public key pairs using the key generation function of a known good implementation and modify five of the public key values so that they are incorrect, leaving five values unchanged (i.e., correct). The evaluator shall obtain in response a set of 10 PASS/FAIL values. Key Generation for Finite-Field Cryptography (FFC) The evaluator shall verify the implementation of the Parameters Generation and the Key Generation for FFC by the TOE using the Parameter Generation and Key Generation test. This test verifies the

ability of the TSF to correctly produce values for the field prime p, the cryptographic prime q (dividing p-1), the cryptographic group generator g, and the calculation of the private key x and public key y. The Parameter generation specifies 2 ways (or methods) to generate the cryptographic prime q and the field prime p:

Cryptographic and Field Primes:

Primes q and p shall both be provable primes

Primes q and field prime p shall both be probable primes

and two ways to generate the cryptographic group generator g:

Cryptographic Group Generator:

Generator g constructed through a verifiable process

Generator g constructed through an unverifiable process.

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

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

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

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

g not= 0,1

q divides p-1

g^q mod p = 1

g^x mod p = y

for each FFC parameter set and key pair.

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

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

(TD0945 applied)

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

2.1.2 Password Conditioning - per TD0865 (ASPP14:FCS_CKM_EXT.1/PBKDF)

2.1.2.1 ASPP14:FCS_CKM_EXT.1.1/PBKDF

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.2.2 ASPP14:FCS_CKM_EXT.1.2/PBKDF

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: Support for PBKDF: The evaluator shall examine the password hierarchy TSS to ensure that the formation of all password based derived keys is described and that the key sizes match that described by the ST author. The evaluator shall check that the TSS describes the method by which the password/passphrase is first encoded and then fed to the SHA algorithm. The settings for the algorithm (padding, blocking, etc.) shall be described, and the evaluator shall verify that these are supported by the selections in this component as well as the selections concerning the hash function itself. The evaluator shall verify that the TSS contains a description of how the output of the hash function is used to form the submask that will be input into the function. For the NIST SP 800-132-based conditioning of the password/passphrase, the required evaluation activities will be performed when doing the evaluation activities for the appropriate requirements (FCS_COP.1.1/KeyedHash). No explicit testing of the formation of the submask from the input password is required. FCS_CKM_EXT.1.1/PBKDF: The ST author shall provide a description in the TSS regarding the salt generation. The evaluator shall confirm that the salt is generated using an RBG described in FCS_RBG_EXT.1.

Section 6.1 of [ST] states the TOE accepts a file-encryption passphrase for authentication of the Data at Rest Protection (DARP) service. This passphrase is conditioned by Data at Rest Protection (DARP) into a Session Key (KEK) value which is used to encrypt the FEK so that it may be written to disk in the encrypted file metadata. The Libgcrypt backend conditions the file-encryption passphrase by producing a new randomized 16-byte salt from the libgrcypt internal DRBG, appends the salt to the binary passphrase value, and runs it through a 310,000 iteration PBKDFv2 w/ SHA384 function. The resulting 32-byte value is then used as an AES secret key to encrypt all FEKs in CBC mode, which are unique to the specific instance of the DARP process.

The passphrase is also required for decrypting any sensitive data through the command-line decryption interface. Upon providing the file-encryption passphrase, the decryption utility will utilize the same Libgcrypt PBKDFv2 w/ SHA384 derivation function used by the DARP service in order to rederive the Session Key. The Session key is then used to decrypt the encrypted FEK from the file metadata and decrypt the file and access the sensitive data.

The TOE also relies on a separate WebUI server certificate passphrase to protect the web server's credentials at rest. Upon start of the TOE service, the TOE requires the user to enter a passphrase before the WebUI service is launched. This passphrase is conditioned with 310,000 iterations of PBKDFv2 w/SHA384 to produce a secret AES key, which is used to decrypt the server private key using AES in CBC mode. The WebUI PBKDF is implemented in the TOE's OpenSSL library which is also responsible for the TLS cryptography.

The evaluator verified this description against the key management hierarchy and has verified that all key sizes are maintained/generated as described.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

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

2.1.3.1 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 [ST] the TOE DARP service is responsible for generating 256-bit AES FEK values to encrypt individual files. All FEKs are generated using DRBG provided by the libgcrypt cryptolibrary and are unique per instance of the DARP service.

Once a FEK is generated and used to encrypt a file, the FEK is encrypted with the Session Key (KEK) and saved to the file metadata.

The TOE generates a single FEK per DARP process instance and reuses the value until the process is restarted. This means that the TOE uses a unique FEK for a set of files within the same session.

The TOE generates 128 and 256-bit AES keys during the TLS handshake. The TOE uses its OpenSSL library to generate the random values used during these connections.

All DRBGs leveraged by the TOE seed at least 256 bits of entropy directly from hardware sources (Intel CPU processors with RDSEED) to ensure that all cryptographic operations that require random data can be initialized with at least 256 bits of security strength.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.1.4 Cryptographic Key Establishment (ASPP14:FCS_CKM.2)

2.1.4.1 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 [ST] states the TOE supports EC-based key establishment (using curves P-256, P-384, and P-521) as part of HTTPS/TLS. The TOE acts as a server for TLS or HTTPS when communicating with the administrators via the Web UI. The TOE's key exchange mechanism is used in the TLS handshake process.

The evaluator confirmed these key establishment schemes corresponded to the key generation schemes in FCS_CKM.1.1/AK.

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

Section 1.3 of AGD states The TOE implements CAVP-certified cryptographic algorithms which are verified as a part of this evaluation. No configuration is needed for the cryptographic modules in order to be compliant. The use of any other cryptographic components beyond those that are identified are neither evaluated nor tested during the TOE's Common Criteria evaluation.

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.

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

2.1.5.1 ASPP14:FCS_CKM_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.1 of [ST] states the TOE's asymmetric key generation is implemented and restricted to the TOE's OpenSSL library which is responsible for cryptography in its WebUI webserver.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.1.6 File Encryption Key (FEK) Generation (FE10:FCS_CKM_EXT.2)

2.1.6.1 FE10:FCS_CKM_EXT.2.1

TSS Assurance Activities: The evaluator shall review the TSS to determine that a description covering how and when the FEKs are generated exists. The description must cover all environments on which the TOE is claiming conformance, and include any preconditions that must exist in order to successfully generate the FEKs. The evaluator shall verify that the description of how the FEKs are generated is consistent with the instructions in the AGD guidance, and any differences that arise from different platforms are taken into account.

Conditional: If 'using a Random Bit Generator' was selected, the evaluator shall verify that the TSS describes how the functionality described by FCS_RBG_EXT.1 (from the [AppPP]) is invoked to generate FEK. To the extent possible from the description of the RBG functionality in FCS_RBG_EXT.1 (from [AppPP]), the evaluator shall determine that the key size being requested is identical to the key size and mode to be used for the decryption/encryption of the user data (FCS_COP.1(1)) (from [AppPP]). (Per TD0455)

Conditional: If 'derived from a password/passphrase' is selected, the examination of the TSS section is performed as part of FCS_CKM_EXT.6 evaluation activities.

Section 6.1 of [ST] states the TOE DARP service is responsible for generating 256-bit AES FEK values to encrypt individual files. All FEKs are generated using DRBG provided by the libgcrypt cryptolibrary and are unique per instance of the DARP service. Once a FEK is generated and used to encrypt a file, the FEK is encrypted with the Session Key (KEK) and saved to the file metadata.

The TOE generates a single FEK per DARP process instance and reuses the value until the process is restarted. This means that the TOE uses a unique FEK for a set of files within the same session.

All DRBGs leveraged by the TOE seed at least 256 bits of entropy directly from hardware sources (Intel CPU processors with RDSEED) to ensure that all cryptographic operations that require random data can be initialized with at least 256 bits of security strength.

The evaluator confirmed the key size in FCS_COP.1 is identical to the key size and mode being used for decryption/decryption.

The evaluator confirmed the descriptions of how FEKs are generated are consistent between the AGD and ST.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.6.2 FE10:FCS_CKM_EXT.2.2

TSS Assurance Activities: The evaluator shall verify the TSS describes how a FEK is used for a protected resource and associated with that resource. The evaluator confirms that-per this description-the FEK is unique per resource (file or set of files) and that the FEK is established using the mechanisms specified in FCS_CKM_EXT.2.1).

Section 6.1 of [ST] states the TOE DARP service is responsible for generating 256-bit AES FEK values to encrypt individual files. All FEKs are generated using DRBG provided by the libgcrypt cryptolibrary and are unique per instance of the DARP service. Once a FEK is generated and used to encrypt a file, the FEK is encrypted with the Session Key (KEK) and saved to the file metadata.

The TOE generates a single FEK per DARP process instance and reuses the value until the process is restarted. This means that the TOE uses a unique FEK for a set of files within the same session.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: The evaluator shall review the instructions in the AGD guidance to determine that any explicit actions that need to be taken by the user to establish a FEK exist-taking into account any differences that arise from different platforms-and are consistent with the description in the TSS.

Section 3.4.1 of AGD describes the FEK is established upon startup or restart of either the DARP service or file encryption.

The TOE includes no other platforms for this evaluation.

Component Testing Assurance Activities: None Defined

2.1.7 Key Encrypting Key (KEK) Support (FE10:FCS_CKM_EXT.3)

2.1.7.1 FE10:FCS_CKM_EXT.3.1

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 determine that a description covering how and when KEK(s) are generated exists. The description must cover all environments on which the TOE is claiming conformance, and include any preconditions that must exist in order to successfully generate the KEKs. The evaluator shall verify that the description of how the KEK(s) are generated is consistent with the instructions in the AGD guidance, and any differences that arise from different platforms are taken into account.

Conditional: If using a RBG was selected the evaluator shall examine the TSS and verify that it describes how the functionality described by FCS_RBG_EXT.1 (from the [AppPP]) is invoked to generate KEK(s). To the extent possible from the description of the RBG functionality in FCS_RBG_EXT.1 (from [AppPP]), the evaluator shall determine that the key size being requested is identical to the key size selected.

Conditional: If derived from a password/passphrase is selected the examination of the TSS section is performed as part of FCS_CKM_EXT.6 evaluation activities.

Section 6.1 of [ST] states the TOE DARP service is responsible for deriving an AES-256 Session Keys (KEK) value from the file-encryption passphrase provided to the DARP service by the administrator. To derive the KEK, the DARP service generates a new random salt value using the libgcrypt DRBG. The random salt is then appended to the binary value of the passphrase. Then the TOE's libgrcrypt library conditions that binary value with PBKDFv2 to produce the shared Session Key which is used to encrypt the FEK. The KEK value itself is never saved to non-volatile memory, however the random salt is saved to the file metadata so that the KEK can be rederived through the decryption tool and decrypt any of the protected files.

Component Guidance Assurance Activities: The evaluator shall review the instructions in the AGD guidance to determine that any explicit actions that need to be taken by the user to establish a KEK exist-taking into account any differences that arise from different platforms-and are consistent with the description in the TSS.

Section 3.3 of AGD describes the explicit actions needed to be taken by the user to start the encryption service. This is consistent with the description in the TSS which details out the inner workings of the keys. A new KEK is established upon startup of the encryption service.

The TOE includes no other platforms for this evaluation.

Component Testing Assurance Activities: None Defined

2.1.8 Cryptographic Key Destruction - per TD0644 (FE10:FCS_CKM_EXT.4)

2.1.8.1 FE10:FCS_CKM_EXT.4.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.8.2 FE10:FCS_CKM_EXT.4.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify the TSS provides a high level description of what it means for keys and key material to be no longer needed and when they should be expected to be destroyed. The evaluator shall verify the TSS provides a high level description of what it means for keys and key material to be no longer needed and when then should be expected to be destroyed.

KMD

The evaluator examines the KMD to ensure it describes how the keys are managed in volatile memory. This description includes details of how each identified key is introduced into volatile memory (e.g. by derivation from user input, or by unwrapping a wrapped key stored in non-volatile memory) and how they are overwritten.

The evaluator shall check to ensure the KMD lists each type of key that is stored in in non-volatile memory, and identifies how the TOE interacts with the underlying platform to manage keys (e.g., store, retrieve, destroy). The description includes details on the method of how the TOE interacts with the platform, including an identification and description of the interfaces it uses to manage keys (e.g., file system APIs, platform key store APIs).

The evaluator examines the interface description for each different media type to ensure that the interface supports the selection(s) and description in the KMD.

If the ST makes use of the open assignment and fills in the type of pattern that is used, the evaluator examines the KMD to ensure it describes how that pattern is obtained and used. The evaluator shall verify that the pattern does not contain any CSPs.

The evaluator shall check that the KMD identifies any configurations or circumstances that may not strictly conform to the key destruction requirement.

If the selection 'destruction of all KEKs protecting target key, where none of the KEKs protecting the target key are derived' is included the evaluator shall examine the TOE's keychain in the KMD and identify each instance when a key is destroyed by this method. In each instance the evaluator shall verify all keys capable of decrypting the target key are destroyed in accordance with a specified key destruction method in FCS_CKM_EXT.4.1. The evaluator shall verify that all of the keys capable of decrypting the target key are not able to be derived to reestablish the keychain after their destruction.

The evaluator shall verify the KMD includes a description of the areas where keys and key material reside and when the keys and key material are no longer needed.

The evaluator shall verify the KMD includes a key lifecycle, that includes a description where key material reside, how the key material is used, how it is determined that keys and key material are no longer needed, and how the material is destroyed once it is not needed and that the documentation in the KMD follows FCS_CKM_EXT.4.1 for the destruction.

Section 6.1 of the ST explains the file-encryption passphrase is cleared from memory once the KEK is derived. The KEK is cleared from memory when encryption service is disabled and cleared from memory during the decryption process once the FEK is derived. The FEK also is cleared from memory when the either the encryption service is disabled or upon decryption of file data.

Section 6.1 of the ST acts as the KMD providing the following information about each key (specifically under subsections for FCS_CKM_EXT.4 and Figure 1): Name, type, size, storage location, and when/how the key is cleared. All information is consistent with the requirements. In addition, the evaluator found the same subsection contained similar language for the AGD documented conditions that may not strictly conform to key destruction.

Component Guidance Assurance Activities: There are a variety of concerns that may prevent or delay key destruction in some cases.

The evaluator shall check that the guidance documentation identifies configurations or circumstances that may not strictly conform to the key destruction requirement, and that this description is consistent with the relevant parts of the TSS and any other relevant Required Supplementary Information.

The evaluator shall check that the guidance documentation provides guidance on situations where key destruction may be delayed at the physical layer and how such situations can be avoided or mitigated if possible.

Some examples of what is expected to be in the documentation are provided here.

When the TOE does not have full access to the physical memory, it is possible that the storage may be implementing wearleveling and garbage collection. This may create additional copies of the key that are logically inaccessible but persist physically. In this case, to mitigate this the drive should support the TRIM command and implements garbage collection to destroy these persistent copies when not actively engaged in other tasks.

Drive vendors implement garbage collection in a variety of different ways, as such there is a variable amount of time until data is truly removed from these solutions. There is a risk that data may persist for a longer amount of time if it is contained in a block with other data not ready for erasure. To reduce this risk, the operating system and file system of the OE should support TRIM, instructing the non-volatile memory to erase copies via garbage collection upon their deletion. If a RAID array is being used, only set-ups that support TRIM are utilized. If the drive is connected via PCI-Express, the operating system supports TRIM over that channel.

The drive should be healthy and contains minimal corrupted data and should be end of lifed before a significant amount of damage to drive health occurs, this minimizes the risk that small amounts of potentially recoverable data may remain in damaged areas of the drive.

Section 4.4 of the AGD describes the operating system's support of TRIM and garbage collection mechanisms through the SSD. It is also explained that the end user should avoid passing in key values to operations outside of the TOE boundary.

Component Testing Assurance Activities: These tests are only for key destruction provided by the application, test 2 does not apply to any keys using the selection 'new value of a key':

Test 1: [Conditional; applies when the application does not perform the zeroization (ie. garbage collecting) for each key held in volatile memory for FCS_CKM_EXT.4.1 (assuming the selection 'destruction of the reference followed by a request for garbage collection')] Applied to each key held in volatile memory and subject to destruction by overwrite by the TOE (whether or not the value is subsequently encrypted for storage in volatile or non-volatile memory). In the case where the only selection made for the key destruction method was removal of power, then this test is unnecessary.

The evaluator shall:

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

2. Cause the cause the TOE or the underlying platform to dump to perform a normal cryptographic processing with the key from Step #1.

3. Cause the TOE to clear the key.

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

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

6. Search the content of the binary file created in Step #5 for instances of the known key value from Step #1.

Steps #1-6 ensure that the complete key does not exist anywhere in volatile memory. If a copy is found, then the test fails.

Test 2: [Conditional; applies when instructing the underlying platform to destroy the key] If new value of a key is selected this test does not apply.

Applied to each key held in non-volatile memory and subject to destruction by the TOE.

The evaluator shall use special tools (as needed), provided by the TOE developer if necessary, to ensure the tests function as intended.

1. Identify the purpose of the key and what access should fail when it is deleted. (e.g. the file encryption key being deleted would cause data decryption to fail.)

2. Cause the TOE to clear the key.

3. Have the TOE attempt the functionality that the cleared key would be necessary for.

4. The test succeeds if Step #3 fails.

Tests 3 and 4 do not apply for the selection instructing the underlying platform to destroy the representation of the key, as the TOE has no visibility into the inner workings and completely relies on the underlying platform.

Test 3: Applied to each key held in non-volatile memory and subject to destruction by overwrite by the TOE. The evaluator shall use a tool that provides a logical view of the media (e.g., MBR file system):

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

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

3. Cause the TOE to clear the key.

4. Search the logical view that the key was stored in for instances of the known key value from Step #1. If a copy is found, then the test fails.

Test 4: Applied to each key held in non-volatile memory and subject to destruction by overwrite by the TOE. The evaluator shall use a tool that provides a logical view of the media:

1. Record the logical storage location of the key in the TOE subject to clearing.

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

3. Cause the TOE to clear the key.

4. Read the logical storage location in Step #1 of non-volatile memory to ensure the appropriate pattern is utilized.

The test succeeds if correct pattern is used to overwrite the key in the memory location. If the pattern is not found the test fails.

Test 1 - The evaluator encrypted and decrypted data using a unique file encryption (FE) passphrase. Once the process completed, the evaluator performed a memory dump of volatile memory and attempted to search the memory dump for the FE passphrase as well as the KEK and DEK. The evaluator confirmed that none of the secret values were found while searching the memory dump.

Test 2 - The evaluator encrypted and decrypted data (internally using the FEK). The evaluator then destroyed the file containing the encrypted FEK. The evaluator attempted to decrypt the file and confirmed the TOE could no longer decrypt the file as it displayed an error.

Test 3 - Not Applicable, the underlying platform is responsible for clearing all keys held in non-volatile memory.

Test 4 - Not Applicable, the underlying platform is responsible for clearing all keys held in non-volatile memory.

2.1.9 Cryptographic Password/Passphrase Conditioning (FE10:FCS_CKM_EXT.6)

2.1.9.1 FE10:FCS_CKM_EXT.6.1

TSS Assurance Activities: There are two aspects of this component that require evaluation: passwords/passphrases of the length specified in the requirement (at least 64 characters or a length defined by the platform) are supported, and that the characters that are input are subject to the selected conditioning function. These activities are separately addressed in the text below.

Support for minimum length: The evaluators shall check to ensure that the TSS describes the allowable ranges for password/passphrase lengths, and that at least 64 characters or a length defined by the platform may be specified by the user.

Support for character set: The evaluator shall check to ensure that the TSS describes the allowable character set and that it contains the characters listed in the SFR.

Support for PBKDF: The evaluator shall examine the TSS to ensure that the formation of all KEKs or FEKs (as decided in the FCS_CKM_EXT.3 selection) is described and that the key sizes match that described by the ST author.

The evaluator shall check that the TSS describes the method by which the password/passphrase is first encoded and then fed to the SHA algorithm. The settings for the algorithm (padding, blocking, etc.) shall be described, and the evaluator shall verify that these are supported by the selections in this component as well as the selections concerning the hash function itself. The evaluator shall verify that the TSS contains a description of how the output of the hash function is used to form the submask that will be input into the function and is the same length as the KEK as specified in FCS_CKM_EXT.4.

For the NIST SP 800-132-based conditioning of the password/passphrase, the required evaluation activities will be performed when doing the evaluation activities for the appropriate requirements (FCS_COP.1.1(4)). If any manipulation of the key is performed in forming the submask that will be used to form the FEK or KEK, that process shall be described in the TSS.

No explicit testing of the formation of the submask from the input password is required.

Section 6.1 of the [ST] states the TOE allows for file-encryption passphrases between 32 and (up to) 128 characters in length. Passphrases may consist of any ACSII-printable character. This passphrase is conditioned with PBKDFv2 w/ SHA-384 key derivation function and DRBG-generated salt to derive the appropriate session key (KEK) from the authentication factor to securely encrypt all FEKs written to memory. Additional TSS descriptions are provided with the associated PBKDF SFRs.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.9.2 FE10:FCS_CKM_EXT.6.2

TSS Assurance Activities: The ST author shall provide a description in the TSS regarding the salt generation. The evaluator shall confirm that the salt is generated using an RBG described in FCS_RBG_EXT.1 (from the [AppPP]).

Section 6.1 of [ST] (under related information for FE10:FCS_CKM_EXT.3 which discusses KEK generation) states to derive the KEK, the DARP service generates a new random salt value using the libgcrypt DRBG. The random salt is then appended to the binary value of the passphrase.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.9.3 FE10:FCS_CKM_EXT.6.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.9.4 FE10:FCS_CKM_EXT.6.4

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.9.5 FE10:FCS_CKM_EXT.6.5

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: Support for minimum length: The evaluators shall check the Operational Guidance to determine that there are instructions on how to generate large passwords/passphrases, and instructions on how to configure the password/passphrase length to provide entropy commensurate with the keys that the authorization factor is protecting.

Section 4.1 of the AGD describes the password complexity requirements for the authorization factor responsible for providing entropy for cryptographic keys.

Component Testing Assurance Activities: Support for Password/Passphrase characteristics: In addition to the analysis above, the evaluator shall also perform the following tests on a TOE configured according to the Operational Guidance:

Test 1: Ensure that the TOE supports password/passphrase lengths as defined in the SFR assignments.

Test 2: Ensure that the TOE does not accept more than the maximum number of characters specified in FCS_CKM_EXT.6.1.

Test 3: Ensure that the TOE does not accept less than the minimum number of characters specified in FCS_CKM_EXT.6.4. If the minimum length is settable by the administrator, the evaluator determines the minimum length or lengths to test.

Test 4: Ensure that the TOE supports passwords consisting of all characters listed in FCS_CKM_EXT.6.2.

Conditioning: No explicit testing of the formation of the authorization factor from the input password/passphrase is required.

Test 1 - The evaluator configured FE passphrase with the minimum and maximum lengths, 32 and 128 respectively. The evaluator confirmed that file encryption successfully started without error in both instances.

Test 2 - The evaluator configured a FE passphrase with one character longer than the maximum acceptable size (129). The evaluator confirmed that file encryption rejected any attempt to start with this passphrase.

Test 3 - The evaluator configured a FE passphrase with one character shorter than the minimum acceptable size (31). The evaluator confirmed that file encryption rejected any attempt to start with this passphrase.

Test 4 - The evaluator configured a FE passphrase consisting of every ASCII printable character. The evaluator confirmed that file encryption successfully started without error.

2.1.10 Cryptographic operation (Key Encryption) (FE10:FCS_COP.1(7))

2.1.10.1 FE10:FCS_COP.1.1(7)

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: Requirement met by the platform

If the platform provides the FEK encryption/decryption, then the evaluator shall examine the TSS to verify that it describes how the FEK encryption/decryption is invoked.

Requirement met by the TOE

The evaluator shall verify the TSS includes a description of the key size used for encryption and the mode used for the key encryption.

Section 6.1 of [ST] states upon file encryption, the TOE DARP service will encrypt an individual FEK using the Session Key (KEK) and write the encrypted FEK to the file metadata. The file-encryption passphrase and the Session Key are never written to disk, but rather are rederived from the passphrase provided upon decryption. The TOE DARP service does not decrypt in-place, but rather decrypts files to a new file location. As a result, the TOE can destroy the encrypted copy of the FEK by destroying the file that stores the value and plaintext data.

In section 6.1 of [ST], Table 3 states that AES-256 CBC is used for key encryption/decryption.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The evaluation activity tests specified for AES in CBC mode in FCS_COP.1.1(1) in the underlying [AppPP] shall be performed.

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

2.1.11 Cryptographic Operation - Hashing (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 [ST] states the TOE's Libgcrypt library provides the SHA-384 algorithm for its use in PBKDFv2 in the respective passphrase verification and KEK derivation actions within the DARP service.

Additionally, the TOE's OpenSSL library provides the SHA-384 used for PBKDFv2 to condition the passphrase needed to decrypt the webserver private key to start the WebUI service.

The TOE uses SHA-256 and SHA-384 hashing when generating TLS server authentication signatures for connections made against its HTTPS WebUI interface.

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.

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.

2.1.13 Cryptographic Operation - Signing - per TD0717 & TD0945 (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 (TD0945 applied)

Test 1: ECDSA FIPS 186-4 or FIPS 186-5 Signature Generation Test. For each supported NIST curve (i.e., P-256, P-384 and P-521) and SHA function pair, the evaluator shall generate 10 1024-bit long messages and obtain for each message a public key and the resulting signature values R and S. To determine correctness, the evaluator shall use the signature verification function of a known good implementation.

Test 2: ECDSA FIPS 186-4 or FIPS 186-5 Signature Verification Test. For each supported NIST curve (i.e., P-256, P-384 and P-521) and SHA function pair, the evaluator shall generate a set of 10 1024-bit message, public key and signature tuples and modify one of the values (message, public key or signature) in five of the 10 tuples. The evaluator shall obtain in response a set of 10 PASS/FAIL values.

RSA Signature Algorithm Tests

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

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

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

2.1.14 Cryptographic Operation - Encryption/Decryption (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 1.3 of AGD states the TOE implements CAVP-certified cryptographic algorithms which are verified as a part of this evaluation. No configuration is needed for the cryptographic modules in order to be compliant. The use of any other cryptographic components beyond those that are identified are neither evaluated nor tested during the TOE's Common Criteria evaluation.

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.

2.1.15 HTTPS Protocol - per TD0736 (ASPP14:FCS_HTTPS_EXT.1/Server)

2.1.15.1 ASPP14:FCS_HTTPS_EXT.1.1/Server

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

Section 6.1 of [ST] states the TOE implements a HTTPS Webserver to provide an administrative Web UI used to manage the TOE. The HTTPS server is RFC2818 compliant which requires the web interface to effectively serve HTTP responses over TLS. As a result, the TOE is also evaluated against the Functional Package for TLS. The TOE webserver binds to the platform's local address on the primary port 32713 and uses an administrator configured server certificate to authenticate itself to HTTPS clients. The TOE does feature a secondary access port, intended for compatibility with other Shift5 containers, however this maps to this same internal process and any difference between external port traffic is indistinguishable to the TOE.

Guidance Assurance Activities: None Defined

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

The evaluator analyzed the traffic collected in ASPP14:FDP_NET_EXT.1-t1 where they confirmed a successful connection and that the traffic is protected with TLS.

2.1.15.2 ASPP14:FCS_HTTPS_EXT.1.2/Server

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

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

This test is performed in conjunction with other TLS testing.

2.1.15.3 ASPP14:FCS_HTTPS_EXT.1.3/SERVER

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: *Other tests are performed in conjunction with the TLS Functional Package, FCS_HTTPS_EXT.2 (dependent on selections in FTP_DIT_EXT.1), and FIA_X509_EXT.1.* (TD0736 applied)

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

This test is performed in conjunction with other TLS testing.

2.1.16 INITIALIZATION VECTOR GENERATION (FE10:FCS_IV_EXT.1)

2.1.16.1 FE10:FCS_IV_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 ensure the TSS describes how nonces are created uniquely and how IVs and tweaks are handled (based on the AES mode). The evaluator shall confirm that the nonces are unique and the IVs and tweaks meet the stated requirements.

Section 6.1 of [ST] states the TOE uses IVs as a part of AES CBC key encryption and tweak values as a part of AES XTS data encryption. IVs used as part of AES in CBC mode are generated from the DRBG in OpenSSL, the same

cryptographic library that is performing the passphrase conditioning needed to decrypt the WebUI's webserver private key. The TOE uses ESSIV calculations based on the file information for AES XTS tweaks.

The TOE also uses AES-GCM ciphersuites for TLS. For these, the TOE derives an IV from shared values in the TLS key block and a monotonically increasing counter to ensure that values are unique and non-repeating.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.1.17 Cryptographic Key Derivation Function (FE10:FCS_KDF_EXT.1)

2.1.17.1 FE10:FCS_KDF_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 the TSS includes a description of the key derivation function and shall verify the key derivation uses an approved derivation mode and key expansion algorithm according to SP 800-108 and SP 800-132.

Section 6.1 of [ST] states the TOE uses 800-132 (PBKDFv2) with HMAC-SHA-384 and 310,000 iterations and a 16-byte salt to transform the administrator's file-encryption passphrase into a Session key used to encrypt and protect individual FEKs.

The TOE also uses 800-132 (PBKDFv2) with HMAC-SHA-384 and 310,000 iterations and a 16-byte salt to transform the WebUI's webserver credential passphrase into a key that can be used to AES-CBC decrypt the saved encrypted WebUI's webserver private key value upon start up.

The number of iterations is determined based on the appropriate delay occurred by the system, however the TOE also features a constant time comparison for derived-key validation to counter timing attacks against the user's passphrase.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.1.18 Key Chaining and Key Storage (FE10:FCS_KYC_EXT.1)

2.1.18.1 FE10:FCS_KYC_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 the TSS contains a high level description of all keychains and authorization methods selected in FIA_AUT_EXT.1 that are used to protect the KEK or FEK.

KMD

The evaluator shall examine the KMD to ensure it describes each key chain in detail, and these descriptions correspond with the selections of the requirement. The description of each key chain shall be reviewed to ensure the options for maintaining the key chain are documented.

The evaluator shall verify the KMD to ensure that it describes how each key chain process functions, such that it does not expose any material that might compromise any key in the chain. A high-level description should include a diagram illustrating the keychain(s) implemented and detail where all keys and keying material is stored or how the keys or key material are derived. The evaluator shall examine the primary key chain to ensure that at no point the chain could be broken without a cryptographic exhaust or knowledge of the KEK or FEK and the effective strength of the FEK is maintained throughout the Key Chain as specified in the requirement.

Section 6.1 of [ST] states the TOE relies on a PBKDFv2-conditioned passphrase for use as a Session Key (KEK). This Session key is used to encrypt the FEK, which are generated per-session. Session Keys and plaintext FEKs are not written to disk, however the encrypted FEK is written to file metadata so that the decryption utility can use the file-encryption passphrase to rederive the Session Key and decrypt the FEK and sensitive data.

The TOE contains no other supplemental keychains that are used to protect a key or keys in the primary keychain. The TOE does feature other keychains, notable a webserver certificate that is stored AES-CBC encrypted and protected by a second, webserver credential passphrase, however this is completely disjoint. Access to the web server credential only allows access to the WebUI, however all data at rest protection of user designated files are only controlled by the file-encryption passphrase.

Figure 1 within section 6.1 of [ST] includes a diagram illustrating the keychain's implementation and key derivation. The evaluator analyzed those diagrams to ensure the keychain was properly protected and all claimed strengths (256-bits) are maintained throughout the lifetime of the keychain. Below Figure 1, the ST explains the TOE uses 256-bit keys through its keychain. The TOE derives 256-bit keys using PBKDFv2, uses AES-256 CBC keys for key encryption, and finally uses AES-256 XTS for file encryption decryption.

Note that other parts of the TOE, which lie outside of the above keychain, can use other size keys. The TOE's TLS server (for administrative access) offers both AES-128 and AES-256 ciphers and the TOE's firmware update images are signed with ECDSA P-384.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

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

2.1.19.1 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 [ST] states the TOE includes two cryptographic libraries as noted in the tables above which detail out the cryptographic algorithm certificates. The TOE relies on either an AES-256 CTR DRBG with a derivation function for OpenSSL or a SHA256-based HMAC_DRBG for libgcrypt. Both DRBGs seed with at least 256-bits of entropy from a direct link to the hardware-based noise source of the host platform, which in the evaluated configuration relies on Intel CPUs with RDSEED instructions.

The evaluator confirmed that the additional elements from FCS_RBG_EXT.2 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).

The TOE implements DRBG functionality so this requirement is not applicable.

2.1.20 Random Bit Generation from Application - per TD0931 (ASPP14:FCS_RBG_EXT.2)

2.1.20.1 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 - 14). The next three are entropy input, nonce, and personalization string for the instantiate operation. The next two are additional input and entropy input for the first call to generate. The final two are additional input and entropy input for the second call to generate. These values are randomly generated. 'generate one block of random bits' means to generate random bits with number of returned bits equal to the Output Block Length (as defined in NIST SP 800-90A).

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

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

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

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

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

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

2.1.20.2 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.

Entropy documentation has been provided to NIAP. Additionally, section 6.1 of the ST summarizes the DRGB functionality.

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.

No further testing activities are currently required.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

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

2.1.21.1 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 [ST] states the TOE relies on a file-encryption passphrase used to protect the Data at Rest Protection, a passphrase to access the WebUI, a WebUI webserver private key, and a passphrase to encrypt/decrypt the webserver private key. Only the TOE's webserver private key is stored to non-volatile memory. Upon startup of the WebUI webserver service, the administrator provides the passphrase to decrypt the webserver private key which is conditioned into a 256-bit AES key and used with AES-CBC to perform the decryption so that the webserver can start up. Since the PBKDF-conditioned file-encryption passphrase is used as a KEK, it is never stored to non-volatile storage.

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 only claimed credential that is stored to non-volatile memory is the webserver private key. The evaluator attempted to access the encrypted copy of the webserver certificate and was not successful. The evaluator decrypted the WebUI certificate using openssl and the same PBKDF parameters claimed in the ST for decryption/encryption of the certificate. The evaluator confirmed the decrypted output was a plaintext WebUI certificate.

2.1.22 TLS Protocol (PKGTLS11:FCS_TLS_EXT.1)

2.1.22.1 PKGTLS11:FCS_TLS_EXT.1.1

TSS Assurance Activities: None Defined

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

Section 1.1 of the AGD, Protection of DARP WebUI traffic through TLS as a server is included in the list of evaluated security functions.

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.1.23 TLS Server Protocol - per TD0779, TD0739, TD0726 & TD0442 (PKGTLS11:FCS_TLSS_EXT.1)

2.1.23.1 PKGTLS11:FCS_TLSS_EXT.1.1

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

Section 6.1 of the ST states the TOE acts as a webserver to present an administrator WebUI where the TOE's Data at Rest protection can be managed. The ciphersuites for the WebUI are addressed in the claims for PKGTLS11:FCS_TLSS_EXT.1.1 under Section 5. In the evaluated configuration, the TOE is restricted to these TLSv1.2 ciphersuites and no other previous TLS versions. Any attempt to request a new TLS connection with a restricted ciphersuite or version by a TLS client will be promptly rejected by the TOE. The TOE supports the elliptic curves: NIST P-256, P-384, P-521, and no other curves/parameters/groups.

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

Section 1.3 of AGD states the TOE implements CAVP-certified cryptographic algorithms which are verified as a part of this evaluation. No configuration is needed for the cryptographic modules in order to be compliant. The use of any other cryptographic components beyond those that are identified are neither evaluated nor tested during the TOE's Common Criteria evaluation. The TOE implements compliant TLS protocols which are verified as a part of this evaluation. No additional configuration is needed for the TLS protocol in order to be compliant.

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

Test 1: The evaluator shall establish a TLS connection using each of the cipher suites specified by the requirement. This connection may be established as part of the establishment of a higher-level protocol, e.g., as part of an EAP session. It is sufficient to observe the successful negotiation of a cipher suite to satisfy the intent of the test; it is not necessary to examine the characteristics of the encrypted traffic in an attempt to discern the cipher suite being used (for example, that the cryptographic algorithm is 128-bit AES and not 256-bit AES).

Test 2: The evaluator shall send a Client Hello to the server with a list of cipher suites that does not contain any of the cipher suites in the server's ST and verify that the server denies the connection. Additionally, the evaluator shall send a Client Hello to the server containing only the TLS_NULL_WITH_NULL_NULL cipher suite and verify that the server denies the connection.

Test 3: If RSA key exchange is used in one of the selected ciphersuites, the evaluator shall use a client to send a properly constructed Key Exchange message with a modified EncryptedPreMasterSecret field during the TLS handshake. The evaluator shall verify that the handshake is not completed successfully and no application data flows.

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

Test 4.1: Removed per TD0469

Test 4.2: Modify a byte in the data of the client's Finished handshake message, and verify that the server rejects the connection and does not send any application data.

Test 4.3: Demonstrate that the TOE will not resume a session for which the client failed to complete the handshake (independent of TOE support for session resumption):

Test 4.3i [conditional]: If the TOE does not support session resumption based on session IDs according to RFC4346 (TLS1.1) or RFC5246 (TLS1.2) or session tickets according to RFC5077, the evaluator shall perform the following test:

a) The evaluator shall send a Client Hello with a zero-length session identifier and with a SessionTicket extension containing a zero-length ticket.

b) The evaluator shall verify the server does not send a NewSessionTicket handshake message (at any point in the handshake).

c) The evaluator shall verify the Server Hello message contains a zero-length session identifier or passes the following steps:

Note: The following steps are only performed if the ServerHello message contains a non-zero length SessionID.

d) The evaluator shall complete the TLS handshake and capture the SessionID from the ServerHello.

e) The evaluator shall send a ClientHello containing the SessionID captured in step d). This can be done by keeping the TLS session in step d) open or start a new TLS session using the SessionID captured in step d).

f) The evaluator shall verify the TOE (1) implicitly rejects the SessionID by sending a ServerHello containing a different SessionID and by performing a full handshake (as shown in Figure 1 of RFC 4346 or RFC 5246), or (2) terminates the connection in some way that prevents the flow of application data.

Test 4.3ii [conditional]: If the TOE supports session resumption using session IDs according to RFC4346 (TLS1.1) or RFC5246 (TLS1.2), the evaluator shall carry out the following steps (note that for each of these tests, it is not necessary to perform the test case for each supported version of TLS):

a) The evaluator shall conduct a successful handshake and capture the TOE-generated session ID in the Server Hello message. The evaluator shall then initiate a new TLS connection and send the previously captured session ID to show that the TOE resumed the previous session by responding with ServerHello containing the same SessionID immediately followed by ChangeCipherSpec and Finished messages (as shown in Figure 2 of RFC 4346 or RFC 5246).

b) The evaluator shall initiate a handshake and capture the TOE-generated session ID in the Server Hello message. The evaluator shall then, within the same handshake, generate or force an unencrypted fatal Alert message immediately before the client would otherwise send its ChangeCipherSpec message thereby disrupting the handshake. The evaluator shall then initiate a new Client Hello using the previously captured session ID, and verify that the server (1) implicitly rejects the session ID by sending a ServerHello containing a different SessionID and performing a full handshake (as shown in figure 1 of RFC 4346 or RFC 5246), or (2) terminates the connection in some way that prevents the flow of application data.

Test 4.3iii [conditional]: If the TOE supports session tickets according to RFC5077, the evaluator shall carry out the following steps (note that for each of these tests, it is not necessary to perform the test case for each supported version of TLS):

a) The evaluator shall permit a successful TLS handshake to occur in which a session ticket is exchanged with the non-TOE client. The evaluator shall then attempt to correctly reuse the previous session by sending the session ticket in the ClientHello. The evaluator shall confirm that the TOE successfully resumes the session in accordance with section 3.1 of RFC 5077.

b) The evaluator shall permit a successful TLS handshake to occur in which a session ticket is exchanged with the non-TOE client. The evaluator will then modify the session ticket and send it as part of a new Client Hello message. The evaluator shall confirm that the TOE either (1) implicitly rejects the session ticket by performing a full handshake (as shown in figure 3 or 4 of RFC 5077), or (2) terminates the connection in some way that prevents the flow of application data.

c) The evaluator shall send the TSF a Client Hello with a SessionTicket extension, and observe the TSF responds with a Server Hello with an empty SessionTicket extension. The evaluator shall then send the TSF a invalid Finished message, and observe that the TSF terminates the session without sending a valid newTicket message.

Note: if the TSF sends a newTicket message prior to terminating the session, the evaluator shall confirm the ticket is invalid by attempting to use the ticket to renew the session and observe that the TSF either (1) implicitly rejects the session ticket by performing a full handshake (as shown in figure 3 or 4 of RFC 5077), or (2) terminates the connection in some way that prevents the flow of application data.

(TD0779 applied, supersedes TD0588)

Test 4.4: Send a message consisting of random bytes from the client after the client has issued the ChangeCipherSpec message and verify that the server denies the connection.

Test 1 - The evaluator made a TLS connection using each of the claimed ciphersuites. The evaluator was able to capture each ciphersuite using a packet capture.

Test 2 - The evaluator configured a client to use TLS. The evaluator attempted to require the TLS_NULL_WITH_NULL_NULL ciphersuite and the attempt was rejected. The evaluator then attempted to require a list of ciphersuites not supported by the TOE and the attempt was rejected.

Test 3 - Corrupt_Encrypted_PreMasterSecret is not applicable to the TOE and is therefore omitted.

Test 4.1 - Removed per TD0469.

Test 4.2 - The evaluator configured a client to use TLS. The evaluator then created packets with the required options modified. The connection was denied as expected.

Test 4.3 – The TOE supports functionality for session resumption based on session tickets. As a result, tests 4.3i and 4.3ii do not apply. For Test 4.3iii, the evaluator walked through a successful TLS handshake and generated a session ticket. The evaluator attempted to resume the session with the valid session ticket and found that the TOE successfully resumed the session. The evaluator then attempted a second test with an initial control connection that generated an initial session ticket and then modified the session ticket as a part of the Client Hello message during session resumption. The evaluator observed the TOE implicitly rejected the session ticket by performing a full handshake. Test 4.3iii Part C is performed in conjunction with Test 4.4 as they rely on the same modification to traffic. In this test, the evaluator observed that the TOE did not return a New Session Ticket, but rather terminated the TLS connection with an alert.

Test 4.4 - The evaluator configured a client to use TLS. The evaluator then created packets with the required options modified. The connection was denied as expected.

2.1.23.2 PKGTLS11:FCS_TLSS_EXT.1.2

TSS Assurance Activities: The evaluator shall verify that the TSS contains a description of the denial of old SSL and TLS versions consistent relative to selections in FCS_TLSS_EXT.1.2.

Section 6.1 of [ST] states the ciphersuites for the WebUI are addressed in the claims for PKGTLS11:FCS_TLSS_EXT.1.1 under Section 5. In the evaluated configuration, the TOE is restricted to these TLSv1.2 ciphersuites and no other previous TLS versions. Any attempt to request a new TLS connection with a restricted ciphersuite or version by a TLS client will be promptly rejected by the TOE.

Guidance Assurance Activities: The evaluator shall verify that the AGD guidance includes any configuration necessary to meet this requirement.

Section 1.3 of AGD states the TOE implements CAVP-certified cryptographic algorithms which are verified as a part of this evaluation. No configuration is needed for the cryptographic modules in order to be compliant. The use of

any other cryptographic components beyond those that are identified are neither evaluated nor tested during the TOE's Common Criteria evaluation. The TOE implements compliant TLS protocols which are verified as a part of this evaluation. No additional configuration is needed for the TLS protocol in order to be compliant.

Testing Assurance Activities: Test 1: The evaluator shall send a Client Hello requesting a connection with version SSL 2.0 and verify that the server denies the connection. The evaluator shall repeat this test with SSL 3.0 and TLS 1.0, and TLS 1.1 if it is selected.

The evaluator configured a client to use TLS. The evaluator then attempted to initiate the client's connection using SSL 2.0, SSL 3.0, TLS 1.0, TLS 1.1, and TLS 1.2. The TOE accepted the client connection with TLS 1.2 and rejected every other connection attempt.

2.1.23.3 PKGTLS11:FCS_TLSS_EXT.1.3

TSS Assurance Activities: The evaluator shall verify that the TSS describes the key agreement parameters of the server's Key Exchange message

Section 6.1 of [ST] states the TOE supports the elliptic curves: NIST P-256, P-384, P-521, and no other curves/parameters/groups.

Guidance Assurance Activities: The evaluator shall verify that any configuration guidance necessary to meet the requirement must be contained in the AGD guidance.

Section 1.3 of AGD states the TOE implements CAVP-certified cryptographic algorithms which are verified as a part of this evaluation. No configuration is needed for the cryptographic modules in order to be compliant. The use of any other cryptographic components beyond those that are identified are neither evaluated nor tested during the TOE's Common Criteria evaluation. The TOE implements compliant TLS protocols which are verified as a part of this evaluation. No additional configuration is needed for the TLS protocol in order to be compliant.

Testing Assurance Activities: The evaluator shall conduct the following tests. The testing can be carried out manually with a packet analyzer or with an automated framework that similarly captures such empirical evidence. Note that this testing can be accomplished in conjunction with other testing activities. For each of the following tests, determining that the size matches the expected size is sufficient.

Test 1: [conditional] If RSA-based key establishment is selected, the evaluator shall configure the TOE with a certificate containing a supported RSA size and attempt a connection. The evaluator shall verify that the size used matches that which is configured and that the connection is successfully established. The evaluator shall repeat this test for each supported size of RSA-based key establishment. (TD0739 applied)

Test 2: [conditional] If finite-field (i.e. non-EC) Diffie-Hellman ciphers are selected, the evaluator shall attempt a connection using a Diffie-Hellman key exchange with a supported parameter size or supported group. The evaluator shall verify that the key agreement parameters in the Key Exchange message are the ones configured. The evaluator shall repeat this test for each supported parameter size or group.

Test 3: [conditional] If ECDHE ciphers are selected, the evaluator shall attempt a connection using an ECDHE ciphersuite with a supported curve. The evaluator shall verify that the key agreement parameters in the Key Exchange message are the ones configured. The evaluator shall repeat this test for each supported elliptic curve.

Test 1 Not applicable as RSA-based key establishment is not selected.

Test 2 Not applicable as non-EC Diffie-Hellman ciphers are not selected.

Test 3 The evaluator made a TLS connection with the TOE server using the following ECDHE key exchanges: secp256r1, secp384r1, secp521r1. The evaluator was able to capture each key size using a packet capture.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.1.24 Validation (FE10:FCS_VAL_EXT.1)

2.1.24.1 FE10:FCS_VAL_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.24.2 FE10:FCS_VAL_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: Conditional: If 'validating' is selected in FCS_VAL_EXT.1.1, the evaluator shall examine the TSS to determine which authorization factors

support validation.

The evaluator shall examine the TSS to ensure that it contains a high-level description of how how the submasks are validated. If multiple submasks are used within the TOE, the evaluator shall verify that the TSS describes how each is validated (e.g., each submask validated before combining, once combined validation takes place).

Conditional: If 'receiving assertion' is selected in FCS_VAL_EXT.1.1, the evaluator shall examine the TSS to verify that it describes the enviroments that can be leveraged with the TOE and how each claims to perform validation. The evaluator shall ensure that none of the stated platform validation mechanisms weaken the key chain of the product.

In section 6.1 of [ST] Validation of the user's file-encryption passphrase only occurs upon decryption. The TOE relies on ephemeral file-encryption passwords, meaning the passphrase is set per-instance of the DARP utility rather than set globally and rechecked. This allows the TOE to generate different passphrase conditioning salts, and thus different KEKs, per reboot of the TOE application even when the same passphrase is used.

Upon starting the encryption service, the TOE generates a Key Confirmation Value (KCV) which serves as a stored hash of the user's password. The KCV is calculated using a similar process as the session key using PBKDF2 w/ SHA-384 with different parameters set for its input salt and iteration count and stored within the encrypted file payload. The KCV is cryptographically based on SHA-384 and is used as a stored comparison hash with the added benefit of dramatically slowing any offline brute-force or dictionary attacks.

Upon decryption, the TOE validates the user's file-encryption passphrase by first pulling the KCV metadata from the referenced file and re-deriving the stored KCV value. Provided the DARP service stored fingerprint matches, the DARP service will proceed to derive the necessary keys for file decryption.

Component Guidance Assurance Activities: If the validation functionality is configurable, the evaluator shall examine the operational guidance to ensure it describes how to configure the TOE to ensure the limits regarding validation attempts can be established.

Section 4.1 of AGD explains the TOE's validation functionality is not configurable.

Component Testing Assurance Activities: None Defined

2.1.25 Validation Remediation (FE10:FCS_VAL_EXT.2)

2.1.25.1 FE10:FCS_VAL_EXT.2.1

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 determine which remediation options are supported for which authentication options.

Section 6.1 of [ST] states the derivation and validation of the fingerprint is done with a constant time comparison, ensuring that any attempt requires a pre-determinized minimal comparison time of 10 seconds. At a maximum, the TOE can only perform 8,460 attempts to validate the file-encryption passphrase per 24-hour period. The file encryption passphrase is the only authentication supported for access to the Data-at-Rest protection and for validation remediation.

Component Guidance Assurance Activities: If the remediation functionality is configurable, the evaluator shall examine the operational guidance to ensure it describes how to configure the TOE to ensure the limits regarding validation attempts can be established.

Section 4.1 of AGD explains the TOE's password validation limits are not configurable. Validations are strictly limited to a maximum of 8,640 attempts per 24-hour period due to constant-time validation comparisons.

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

Test 1: The evaluator shall determine the limit on the average rate of the number of consecutive failed authorization attempts. For each authentication factor supported, the evaluator will test the TOE by entering that number of incorrect authorization factors in consecutive attempts to access the protected data. If the limit mechanism includes any 'lockout' period, the time period tested should include at least one such period. Then the evaluator will verify that the TOE behaves as described in the TSS.

The evaluator repeatedly attempted to decrypt using incorrect authorization factors while collecting time stamps in between attempts using a bash command. The evaluator confirmed the instituted delay between authorization attempts was at least the claimed duration of 10 seconds without manual input delay. The evaluator calculated from the number of attempts over the tested time period that it was not possible to exceed 8,460 password attempts within a 24-hour window.

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 [ST] states the TOE implements file-based encryption for protection of sensitive data. Sensitive data includes any data provided to the TOE's DARP service to be encrypted on the removable storage drive. Sensitive data may also include authorization passphrases and keys used to secure any plaintext data, however the FEK is the only file-encryption key written to disk (only after it has been encrypted). The TOE DARP service is started by the user providing a new file-encryption passphrase. Once the passphrase is provided, the DARP service will use the passphrase to derive a session key to use as a KEK and will generate a random 256-bit FEK to encrypt all files. The same KEK / FEK is used by the TOE until the DARP service is restarted. With the DARP service unlocked, the TOE will listen at predefined directory for incoming sensitive data written by external processes. Upon receiving sensitive information, the DARP service will begin using AES-XTS 256-bit encryption to protect the file's contents. Once the file encryption process completes, the encrypted FEK is saved to the metadata of the file. Upon shutdown of the DARP service, all copies of the KEK and FEK are destroyed. The sensitive data can then be retrieved by the decryption utility which takes the user's passphrase, rederives the Session key, and decrypts the

encrypted FEK saved to the file metadata. The decryption utility will decrypt the files contents, saving the decrypted data to a new location and leaving the original, encrypted file in place.

Component Guidance Assurance Activities: None Defined

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

If 'implement functionality to encrypt sensitive data as defined in the PP-Module for File Encryption' or 'protect sensitive data in accordance with FCS_STO_EXT.1' is selected, the evaluator shall inventory the filesystem locations where the application may write data. The evaluator shall run the application and attempt to store sensitive data. The evaluator shall then inspect those areas of the filesystem to note where data was stored (if any), and determine whether it has been encrypted. (TD0756 applied)

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

Platforms: Android....

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

Platforms: Microsoft Windows....

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

Platforms: Apple iOS....

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

Platforms: Linux....

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

Platforms: Oracle Solaris....

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

Platforms: Apple macOS....

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

The TOE implements functionality to encrypt sensitive data as defined in the PP-Module for File Encryption for all sensitive data not covered by FCS_STO_EXT.1.

The evaluator referenced testing done under FE10:FDP_PRT_EXT.1-t1 where the evaluator encrypted data by sending data to the /opt/shift5/data/ramdisk directory. The application then encrypted the written data and stored the output in /opt/shift5/data/incoming. The evaluator searched for evidence of the plaintext pattern in both volatile and non-volatile memory and did not find it, ensuring the file was encrypted.

2.2.2 Access to Platform Resources - per TD0822 (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 2.2 of the AGD states the TOE does utilize network connectivity for its WebUI, however the TOE utilizes no other hardware resources.

Testing Assurance Activities: Platforms: Android....

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

Platforms: Microsoft Windows....

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

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

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

Platforms: Apple iOS....

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

Platforms: Linux....

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

Platforms: Oracle Solaris....

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

Platforms: Apple macOS....

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

(TD0822 applied)

The TOE platform is Linux-based, so the evaluator reviewed the functionality provided by the web interface, the administrator guidance document, and documentation provided by the TSS and concluded that network connectivity was the only hardware resource accessed by the TOE.

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 2.2 of the AGD states the TOE does not access any sensitive information repositories on the platform.

Testing Assurance Activities: Platforms: Android....

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

Platforms: Microsoft Windows....

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

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

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

Platforms: Apple iOS....

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

Platforms: Linux....

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

Platforms: Oracle Solaris....

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

Platforms: Apple macOS....

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

(TD0822 applied)

The evaluator analyzed the TOE software and documents (as discussed under ASPP14:FDP_DEC_EXT.1.1-t1) and did not find any evidence of access to sensitive information repositories, which is consistent with the security claims for the TOE.

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 evaluator ran the application, captured network traffic, and confirmed that only traffic associated with the WebUI interface was found associated with the TOE.

Test 2: The evaluator performed a port scan and verified that the TOE does not open any port aside from documented ports.

2.2.4 Protection of Data in Power Managed States (FE10:FDP_PM_EXT.1)

2.2.4.1 FE10:FDP_PM_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.4.2 FE10:FDP_PM_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.4.3 FE10:FDP_PM_EXT.1.3

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 state(s) that are supported by this capability. For each state, the evaluator ensures that the TSS contains a description of

how the state is entered, and the actions of the TSF on entering the state, specifically addressing how multiple open resources (of each type) are protected, and how keying material associated with these resources is protected (if different from that described elsewhere). The TSF shall also describe how the state is exited, and how the requirements are met during this transition to an operational state.

The evaluator shall verify the TSS provides a description of what keys and key material are destroyed when entering any protected state.

KMD

The evaluator shall verify the KMD includes a description of the areas where keys and key material reside. The evaluator shall verify the KMD includes a key lifecycle that includes a description where key material reside, how the key material is used, and how the material is destroyed once a claimed power state is entered and that the documentation in the KMD follows FCS_CKM_EXT.4.1 for the destruction.

Section 6.2 of [ST] states the TOE provides the compliant power-saving state power-off and G3, mechanical off. The TOE enters this state when the operator shuts off the device. The TOE must be fully rebooted from this state. Upon transition to this state, the TOE will close all open resources and destroy any active key material. These power states can be exited by rebooting the platform. New files can be encrypted by restarting the TOE encryption service which requires supplying all valid credentials. Files can be decrypted again at user request after supplying the decryption password each time.

Section 6.1 of the ST (specifically the subsections to the various FCS_CKM* and FCS_COP* SFRs) acts as the KMD providing the following information about each key (primarily under the FE10:FCS_CKM_EXT.4 section): Name, type, size, storage location, and when/how the key is cleared. All information is consistent with the requirements.

Component Guidance Assurance Activities: The evaluator shall check the Operational Guidance to determine that it describes the states that are supported by the TOE, and provides information related to the correct configuration of these modes and the TOE.

The evaluator shall validate that guidance documentation contains clear warnings and information on conditions in which the TOE may end up in a non-protected state. In that case it must contain mitigation instructions on what to do in such scenarios.

Section 4.2 of the AGD explains the manifold has only two power states: on and off. If the manifold is turned on and the OS login prompt is missing, turn the power off and then turn the power back on. Warning on Non-Protected States: The TOE only protects new incoming data once the DARP service has been started in the WebUI. After any restart of the platform, the TOE is in a non-protected state and must once again be restarted and initialized before it can protect new incoming data. Mitigation Instructions: Following any system reboot or power cycle, the administrator must log into the console, run the initialization scripts (start_fde and start_fe), supply the

WebUI Webserver Certificate Passphrase via the start script, and re-enable fi le encryption through the DARP WebUI..

Component Testing Assurance Activities: The following tests must be performed by the evaluator for each supported State, type of resource, platform, and authorization factor:

Test 1: Following the Operational guidance, configure the Operational Environment and the TOE so that the lower power state of the platform is enabled and protected by the TOE. Open several resources (documented in the test report) that are protected. Invoke the lower power state. On resumption of normal power attempt to access a previously-opened protected resource, observe that an incorrect entry of the authorization factor(s) does not result in access to the system, and that correct entry of the authorization factor(s) does result in access to the resources.

The evaluator encrypted new sensitive data and showed that it could be decrypted after supplying the correct FE passphrase. The evaluator then rebooted the TOE and attempted to re-decrypt the previous file with an incorrect passphrase and confirmed encryption failed and access to the sensitive information was denied. The evaluator then decrypted and accessed the plaintext through correct entry of the passphrase.

2.2.5 Protection of Selected User Data (FE10:FDP_PRT_EXT.1)

2.2.5.1 FE10:FDP_PRT_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.5.2 FE10:FDP_PRT_EXT.1.2

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 determine that it lists each type of resource that can be encrypted (e.g., file, directory) and what 'encrypted' means in terms of the resource (e.g., 'encrypting a directory' means that all of the files contained in the directory are encrypted, but the data in the directory itself (which are filenames and pointers to the files) are not encrypted).

The evaluator shall also confirm that the TSS describes how each type of resource listed is encrypted and decrypted by the TOE. The evaluator shall ensure that this description includes the case where an existing file or set of files is encrypted for the first time; a new file or set of files is created and encrypted; an existing file or set of files is re-encrypted (that is, it had been initially encrypted; it was decrypted (by the TOE) for use by the user, and is then subsequently re-encrypted); and corresponding decryption scenarios. If other scenarios exist due to product implementation/features, the evaluator shall ensure that those scenarios are covered in the TSS as well.

The evaluator shall examine the TSS to ensure there is a high-level description of how the FEK is protected.

The evaluator shall examine the TSS to ensure there is a description of how the FEK is protected.

The evaluator shall examine the TSS to ensure that it describes all temporary files/resources created or memory used during the decryption/encryption process and when those files/resources or memory is no longer needed.

The TSS shall describe how the TSF or TOE platform deletes the non-volatile memory (for example, files) and volatile memory locations after the TSF is done with its decryption/encryption operation.

Section 6.2 of the [ST] states that once unlocked through the administrator WebUI, the DARP service listens in a predefined /opt/shift5/data/incoming directory for new *.shift5 files from external processes. Once a new plaintext file appears, the DARP service AES-XTS 256-bit encrypts the contents of this file using the previously derived FEK. The resulting .s5e output from the DARP service is saved to the /opt/shift5/data/ramdisk directory. To ensure the original plaintext contents of the file are no longer on disk, the DARP service will delete the file while performing an overwrite of zeros to the original file and any internal plaintext data buffers. The TOE does not utilize any additional, temporary resources to perform its encryption and decryption. Once the DARP service retrieves the sensitive information and finishes the encryption process, all filespace resources will be deleted and overwritten. Similarly, any volatile memory buffers used to store sensitive data before it is encrypted will be overwritten with new values and then cleared upon shutdown.

Upon decryption of the file with the decryption utility, the decryption tool will rederive the KEK, use the KEK to decrypt the FEK, use the FEK to decrypt the encrypted data, and save that decrypted data to a new location in the same directory. The original file is never removed by the decryption tool, however the user can rely on platform tools to overwrite the file with pseudorandom data prior to removing the file from the filesystem.

Component Guidance Assurance Activities: If the TOE creates temporary objects and these objects can be protected through administrative measures (e.g., the TOE creates temporary files in a designated directory that can be protected through configuration of its access control permissions), then the evaluator shall check the Operational Guidance to ensure that these measures are described.

If there are special measures necessary to configure the method by which the file or set of files are encrypted (e.g., choice of algorithm used, key size, etc.), then those instructions shall be included in the Operational Guidance and

verified by the evaluator. In these cases, the evaluator checks to ensure that all non-TOE products used to satisfy the requirements of the ST that are described in the Operational Guidance are consistent with those listed in the ST, and those tested by the evaluation activities of this PP-Module.

There are no additional guidance evaluation activities for FDP_PRT_EXT.1.2.

Not applicable, the TOE does not create any temporary objects and there are no special configurations necessary to configure file encryption.

Component Testing Assurance Activities: The evaluator shall also perform the following tests. All instructions for configuring the TOE and each of the environments must be included in the Operational Guidance and used to establish the test configuration.

For each resource and decryption/encryption scenario listed in the TSS, the evaluator shall ensure that the TSF is able to successfully encrypt and decrypt the resource using the following methodology:

Monitor the temporary resources being created (if any) and deleted by the TSF-the tools used to perform the monitoring (e.g., procmon for a Windows system) shall be identified in the test report. The evaluator shall ensure that these resources are consistent with those identified in the TSS, and that they are protected as specified in the Operational Guidance and are deleted when the decryption/encryption operation is completed.

Test 1: This test only applies for application provided functionality.

1. Using a file editor, create and save a text file that is encrypted per the evaluation configured encryption policy. The contents of the file will be limited to a known text pattern to ensure that the text pattern will be present in all encryption/decryption operations performed by the TOE.

2. Exit the file editor so that the file (including its known text pattern) has 'completed the decryption/encryption operation' and process memory containing the known text pattern is released.

3. The evaluator will take a dump of volatile memory and search the retrieved dump for the known pattern. The test fails if the known plaintext pattern is found in the memory dump.

4. The evaluator will search the underlying non-volatile storage for the known pattern. The test fails if the known plaintext pattern is found in the search.

The evaluator used an external text editor to create and save a text file with a recorded string and copied the file over to the TOE filesystem. The evaluator performed encryption and subsequent decryption operations on the text file. The evaluator then performed a memory dump. The evaluator searched for the string in both the memory dump and the TOE's data directories and verified the string was not found.

2.2.6 Destruction of Plaintext Data (FE10:FDP_PRT_EXT.2)

2.2.6.1 FE10:FDP_PRT_EXT.2.1

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 all temporary file (or set of files) that are created in the filesystem of the host during the decryption/encryption process, and that the TSS describes how these files are deleted after the TSF is done with its decryption/encryption operation. Note that if other objects/resources are created on the host that are 1) persistent and 2) visible to other processes (users) on that host that are not filesystem objects, those objects shall be identified and described in the TSS as well.

Section 6.2 of the [ST] states that once unlocked through the administrator WebUI, the DARP service listens in a predefined /opt/shift5/data/incoming directory for new *.shift5 files from external processes. Once a new plaintext file appears, the DARP service AES-XTS 256-bit encrypts the contents of this file using the previously derived FEK. The resulting .s5e output from the DARP service is saved to the /opt/shift5/data/ramdisk directory. To ensure the original plaintext contents of the file are no longer on disk, the DARP service will delete the file while performing an overwrite of zeros to the original file and any internal plaintext data buffers. The TOE does not utilize any additional, temporary resources to perform its encryption and decryption. Once the DARP service retrieves the sensitive information and finishes the encryption process, all filespace resources will be deleted and overwritten. Similarly, any volatile memory buffers used to store sensitive data before it is encrypted will be overwritten with new values and then cleared upon shutdown.

Upon decryption of the file with the decryption utility, the decryption tool will rederive the KEK, use the KEK to decrypt the FEK, use the FEK to decrypt the encrypted data, and save that decrypted data to a new location in the same directory. The original file is never removed by the decryption tool, however the user can rely on platform tools to overwrite the file with pseudorandom data prior to removing the file from the filesystem.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: Test 1: If the TSS creates temporary files/resources during file decryption/encryption, the evaluator shall perform the following tests to verify that the temporary files/resources are destroyed. If the product supported shared files per FIA_FCT_EXT.2, this test must be repeated with a shared file. The evaluator shall use a tool (e.g., procmon for a Windows system) that is capable of monitoring the creation and deletion of files during the decryption/encryption process is performed. A tool that can search the contents of

the hard drive (e.g., winhex) will also be needed. The tools used to perform the monitoring shall be identified in the test report.

(Creating an encrypted document)

Open an editing application.

Create a special string inside the document. The string could be 5-10 words. It is recommended to remove the spaces. This will create a one page document.

Start the file monitoring tool.

Save and close the file.

Encrypt the file using the TOE (if the TOE does not encrypt automatically for the user).

Analysis Steps

If needed, exit/close the TOE.

Stop the file monitoring tool. View the results. Identify any temporary files that were created during the encryption process. Examine to see if the temporary files were destroyed when the TOE closed.

If temporary files remain, these temporary files should be examined to ensure that no plaintext data remains. If plaintext data is found in these files, the test fails.

Search the contents of the hard drive (using the second tool) for the plaintext string used above. (The search should be performed using both ASCII and Unicode formats.)

If the string is found, this means that plaintext from the test fails.

(Creating, encrypting a blank document and then adding text):

Encrypt a blank document using the tool.

Create a special string inside the document. The string could be 5-10 words. It is recommended to remove the spaces. This will create a one page document.

Start the file monitoring tool.

Save and close the file.

Perform the 'Analysis Steps' listed above.

Not Applicable, temporary files/resources are not created during file decryption/encryption.

2.3 IDENTIFICATION AND AUTHENTICATION (FIA)

2.3.1 SUBJECT AUTHORIZATION (FE10:FIA_AUT_EXT.1)

2.3.1.1 FE10:FIA_AUT_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 how user authentication is performed. The evaluator shall verify that the authorization methods listed in the TSS are specified and included in the requirements in the ST.

Requirement met by the TOE:

The evaluator shall first examine the TSS to ensure that the authorization factors specified in the ST are described. For password-based factors the examination of the TSS section is performed as part of FCS_CKM_EXT.6 Evaluation Activities.

Additionally in this case, the evaluator shall verify that the operational guidance discusses the characteristics of external authorization factors (e.g., how the authorization factor must be generated; format(s) or standards that the authorization factor must meet) that are able to be used by the TOE. If other authorization factors are specified, then for each factor, the TSS specifies how the factors are input into the TOE.

Requirement met by the platform:

The evaluator shall examine the TSS to ensure a description is included for how the TOE is invoking the platform functionality and how it is getting an authorization value that has appropriate entropy.

Section 6.3 of [ST] states that the TOE utilizes three separate passphrases: a WebUI access passphrase, a file-encryption passphrase (provided to DARP service through the WebUI), and a webserver credential passphrase. The WebUI passphrase is only used to add an additional layer of access control to the WebUI HTTPS interface, prior to the user providing the file-encryption passphrase to enable the encryption service. This passphrase is not analyzed as part of this evaluation as all user authentication requirements are met by the file-encryption passphrase. Additionally, the WebUI web server credential is stored in a password-based, encrypted manner. This

webserver credential passphrase again only secures the HTTPS private key used to host the HTTPS WebUI which controls access to the WebUI, but not the file encryption key hierarchy.

Since neither of these passphrases are a part of the file encryption key hierarchy used to secure sensitive data under the File Encryption PP, these passphrases do not provide any authentication of user data and only the file-encryption passphrase is analyzed as a part of this requirement.

The file-encryption passphrase is provided by an administrator to the TOE's DARP service through the TOE's WebUI for user authentication, prior to enabling any encryption services. For each new instance of the DARP service, the user configures a new file-encryption passphrase to be used for that session. The DARP service will derive the Session Key (KEK), which is used to protect all FEK values, and generate a new FEK value to be used for all files within this session. Similarly, the decryption process runs through a command line tool which requires the file-encryption passphrase to validate and authenticate the user's passphrase prior to decrypting the sensitive protected data.

Section 6.1 of [ST] states the TOE uses 800-132 (PBKDFv2) with HMAC-SHA-384 and 310,000 iterations and a 16-byte salt to transform the administrator's file-encryption passphrase into a Session key used to encrypt and protect individual FEKs.

Component Guidance Assurance Activities: The evaluator shall verify that the AGD guidance includes instructions for all of the authorization factors. The AGD will discuss the characteristics of external authorization factors (e.g., how the authorization factor is generated; format(s) or standards that the authorization factor must meet, configuration of the TPM device used) that are able to be used by the TOE.

Section 4.1 of the AGD discusses all passwords related to the TOE and the parameters required to configure a valid password for each.

Component Testing Assurance Activities: The evaluator shall ensure that authorization using each selected method is tested during the course of the evaluation, setting up the method as described in the operational guidance and ensuring that authorization is successful and that failure to provide an authorization factor results in denial to access to plaintext data.

[conditional]: If there is more than one authorization factor, ensure that failure to supply a required authorization factor does not result in access to the decrypted plaintext data.

The evaluator tested this in conjunction with testing for FE10:FDP_PM_EXT.1-t1 where the evaluator configured authorization according to operation guidance. In this test, the evaluator then confirmed that failing to provide the

correct authorization factor caused the TOE to deny access to plaintext data. The evaluator confirmed that providing the correct authorization factor resulted in access to plaintext data.

No additional testing is applicable as there is only one claimed authorization factor (password).

2.4 SECURITY MANAGEMENT (FMT)

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

2.4.1.1 ASPP14:FMT_CFG_EXT.1.1

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

Section 6.4 of the ST states the TOE ships with default credentials for accessing the WebUI. In order to initialize the TOE service, an administrator will have to set a new credential value for the WebUI before continuing. For the remaining credentials, does not ship with any default values. Instead, the user must configure a file-encryption passphrase, a passphrase to access the WebUI, a new webserver certificate and private key, and a passphrase to encrypt the webserver private key during the initial provisioning of the TOE. During the initial provisioning of the TOE, the TOE will create a new webserver certificate to use and authenticate web traffic through the WebUI.

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.

Test 1: The TOE only ships with default credentials for the WebUI access passphrase. The remaining credentials were demonstrated to be configured during installation and first startup of the TOE. The evaluator installed and

ran the application and accessed the WebUI using default credentials. The evaluator confirmed that no other functions were accessible until new credentials were assigned.

Test 2: Not applicable as the sole remaining credentials cannot be cleared from the TOE after being assigned.

Test 3: After establishing new credentials as a part of test 1, the evaluator attempted to login using the original default credentials. The evaluator observed the login failed and could no longer access the application.

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 identified /opt as the main branch of the application's data directories. The evaluator then ran find -L . -perm /002 inside the directory and confirmed there were no world writable files.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

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

2.4.2.1 ASPP14:FMT_MEC_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

Section 6.4 of [ST] states the TOE is a containerized-Linux application where all of the configuration options controlled by the application reside within the container's internally mapped /etc/ directory. The TOE does not

feature any configuration options needed to place the device into the evaluated configuration or that affect SFR behavior, however it does support configuration of an optional log level as part of ASPP14:FMT_SMF.1. This configuration option is stored within the platfrom's Kubernetes ConfigMap which utilizes to the host operating systems etcd service, which internally stores value in a key, value database stored in the /var/lib directory. Any remaining configuration option is not intended to be modified by the end user.

Component Guidance Assurance Activities: None Defined

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

Platforms: Android....

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

Platforms: Microsoft Windows....

The evaluator shall determine and verify that Windows Universal Applications use either the Windows.Storage namespace, Windows.UI.ApplicationSettings namespace, or the IsolatedStorageSettings namespace for storing application specific settings. For .NET applications, the evaluator shall determine and verify that the application uses one of the locations listed in https://docs.microsoft.com/en-us/dotnet/framework/configure-apps/ or https://learn.microsoft.com/en-us/previous-versions/dotnet/netframework-4.0/zzdt0e7f(v=vs.100) or https://learn.microsoft.com/en-us/aspnet/core/fundamentals/configuration/ or https://learn.microsoft.com/en-us/aspnet/core/host-and-deploy/iis/web-config for storing application specific (whether application-wide or user-specific) settings. For Classic Desktop applications, the evaluator shall run the application while monitoring it with the SysInternals tool ProcMon and make changes to its configuration. The evaluator shall verify that ProcMon logs show corresponding changes to the the Windows Registry or C:directory. (TD0964 applied, supersedes TD0893)

Platforms: Apple iOS....

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

Platforms: Linux....

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

specific configuration), or /var/lib/ (for configurations controlled by UI and not intended to be directly modified by an administrator).

Platforms: Oracle Solaris....

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

Platforms: Apple macOS....

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

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

The evaluator initialized the TOE and monitored the kubenetes container for the TOE's behavior with strace. The evaluator then accessed the TOE's Kubernetes configuration file and changed the logging level. After reviewing the strace, the evaluator confirmed the changes were stored within the var/lib directory.

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 subsections in Section 3 of the AGD describes the procedure to perform every management function selected in the ST. Each subsection is listed below:

Section 3.1 describes the process to change the container log level.

Section 3.2 describes changing the WebUI webserver credential password.

Section 3.3 describes enabling the encryption service and changing the WebUI access password authentication factor.

Section 3.4.2 describes decryption of selected files.

Section 3.4.3 describes configuration of the file encryption password.

Section 3.5 describes destroying encrypted files.

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

The evaluator performed each management function listed on the ST using guidance documentation:

The evaluator configured the WebUI webserver credential password and file-encryption password authentication factor through initializing the TOE with the start and start_fe scripts.

The evaluator accessed the WebUI and changed WebUI access password authentication factor.

The evaluator accessed the WebUI and enabled the encryption service.

The evaluator destroyed an encrypted file.

The evaluator accessed the TOE's config file and changed the logging level.

The evaluator configured decryption on the TOE and ensured that the TOE could decrypt any encrypted data.

2.4.4 Specification of File Encryption Management Functions (FE10:FMT_SMF.1(2))

2.4.4.1 FE10:FMT_SMF.1.1(2)

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: Conditional Activities: The evaluator shall examine the TSS to ensure that it describes the sequence of activities that take place from an implementation perspective when this activity is performed (for example, how it determines which resources are associated with the KEK, the decryption and re-encryption process), and ensure that the KEK and FEK are not exposed during this change.

Cryptographic Configuration: None for this requirement.

Section 6.4 of [ST] states the TOE is a containerized application that allows management through command line interaction and through the TOE's WebUI. Through the command line interface, an administrator can provide passwords, start the WebUI, manage files, and decrypt files through the decryption utility. Through the WebUI, the administrator can start and stop the encryption service which actively monitors for incoming files to encrypt.

File-encryption passphrases are set per-instance of the DARP service and are changed every time the TOE is restarted. Upon startup of the TOE DARP service, a new password conditioning salt is generated ensuring that a different Session Key (KEK) is used, even if the same passphrase is used. The passphrases used to protect the WebUI access or WebUI webserver certificates can be changed, however, these values do not directly protect any of the cryptographic keys used to encrypt files, but rather just control who can start and stop the TOE DARP service.

The TOE decrypts files to a new location and does not modify the original encrypted payload. In order to perform a cryptographic erase of the encrypted FEK and ciphertext, the TOE instructs the user to overwrite the encrypted file using the host platform's DRBG prior to removing the file from the filesystem. By overwriting the encrypted FEK and ciphertext with pseudorandom data, the sensitive data can no longer be recovered.

Component Guidance Assurance Activities: Conditional Activities: The evaluator shall examine the Operational Guidance to ensure that it describes how the password/passphrase-based authorization factor is to be changed.

Cryptographic Configuration: The evaluator shall determine from the TSS for other requirements (FCS_*, FDP_PRT_EXT, FIA_AUT_EXT) what portions of the cryptographic functionality are configurable. The evaluator shall then review the AGD documentation to determine that there are instructions for manipulating all of the claimed mechanisms.

Section 3.4.3 describes the process of changing the passphrase authorization factor.

None of the cryptographic functionality for the mentioned requirements are configurable.

Component Testing Assurance Activities: Cryptographic Erase: If the TOE uses stored FEKS or KEKs, the evaluator shall examine the key chain to determine that the keys destroyed by a cryptographic erase will result in the data becoming unrecoverable. Testing for this activity is performed for other components in this PP-Module.

The evaluator tested this alongside FE10:FCS_CKM_EXT.4-t2 where the evaluator destroyed the encrypted FEK by overwriting and deleting the encrypted ciphertext where it was stored. The evaluator confirmed that the data was unrecoverable after being deleted.

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 [ST] states the TOE does not transmit any information 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 as the TOE does not transmit PII over the 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. If any explicitly-mapped exceptions are claimed, the evaluator shall check that the TSS identifies these exceptions, describes the static memory mapping that is used, and provides justification for why static memory mapping is appropriate in this case. (TD0798 applied)

Section 6.6 of [ST] states the TOE components are compiled with the '-fstack-protector-strong' flag for native components and the "export CGO_LDFLAGS='-fstack-protector-strong'" flag for all golang applications in order to ensure protection against stack-based buffer overflow protection. The TOE also relies on several precompiled binaries from the Chainguard base container image, all of which are compiled with -fstack-protector-strong. Additionally, all native ELF executables are compiled with the appropriate '-pie' flags (position independent executable) to ensure address space layout randomization is enforced. Golang is a memory-managed language which ensures that all memory allocation locations are randomized, and therefore no additional compilation flags are needed for these executables. The TOE does not use any explicitly-mapped memory regions.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator shall perform either a static or dynamic analysis to determine that no memory mappings are placed at an explicit and consistent address except for any exceptions claimed in the SFR. For these exceptions, the evaluator shall verify that this analysis shows explicit mappings that are consistent with what is claimed in the TSS. The method of doing so varies per platform. For those platforms requiring the same application running on two different systems, the evaluator may alternatively use the same device. After collecting the first instance of mappings, the evaluator must uninstall the application, reboot the device, and reinstall the application to collect the second instance of mappings. (TD0798 applied)

Platforms: Android....

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

Platforms: Microsoft Windows....

The evaluator shall run the same application on two different Windows systems and run a tool that will list all memory mapped addresses for the application. The evaluator shall then verify the two different instances share no

mapping locations. The Microsoft SysInternals tool, VMMap, could be used to view memory addresses of a running application. The evaluator shall use a tool such as Microsoft's BinScope Binary Analyzer to confirm that the application has ASLR enabled.

Platforms: Apple iOS....

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

Platforms: Linux....

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

Platforms: Oracle Solaris....

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

Platforms: Apple macOS....

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

The evaluator installed and ran the application on two different platforms. The evaluator then collected pmap of the TOE on each system and compared the memory mappings using statistical analysis. The results showed that there were no static memory addresses.

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 started with a previously demonstrated decomposition of the TOE into a series of well-understood files and output from a suite of static analysis tools. The evaluator then searched the total findings for any instances of “RWE” in ELF program headers (none found), “WAX” flags in section headers (none found), and any keyword matches to mmap and mprotect. The evaluator found a few instances of similar keywords, so the search was further refined to analyze any instances of the PROT_WRITE/PROT_EXEC permissions, including for various integer values that represent the flags. The evaluator found no conclusive evidence that either incorrect permission combination was ever used.

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

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

Platforms: Apple iOS....

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

Platforms: Linux....

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

Platforms: Oracle Solaris....

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

Platforms: Apple macOS....

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

The evaluator verified SELinux was in enforce mode on the platform used throughout testing. The evaluator confirmed that the TOE ran successfully with this setting.

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 performed normal usage on the TOE through the encryption/decryption of data. The evaluator confirmed the location where user-modifiable files reside does not contain executable files.

2.6.1.5 ASPP14:FPT_AEX_EXT.1.5

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

Section 6.6 of [ST] states the TOE components are compiled with the '-fstack-protector-strong' flag for native components and the "export CGO_LDFLAGS='-fstack-protector-strong'" flag for all golang applications in order to ensure protection against stack-based buffer overflow protection. The TOE also relies on several precompiled binaries from the Chainguard base container image, all of which are compiled with -fstack-protector-strong.

Guidance Assurance Activities: None Defined

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

Platforms: Microsoft Windows....

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

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

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

xor rcx, (...)

call (...)
```

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

Tools such as Canary Detector may help automate these activities.

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

(TD0815 applied)

The evaluator searches the strings and objdump output of all ELF executables for references of the _stack_chk_fail flag using grep. To address the executables without a reference to the _stack_chk_fail flag, the conditional TSS activity was performed to demonstrate that the TOE and its native executables are all compiled with sufficient protections.

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 [ST] includes a list of all platform APIs used in the application.

Component Guidance Assurance Activities: None Defined

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

The evaluator crossreferenced the list of APIs provided in the TSS with the Go developer website https://pkg.go.dev/, and github and verified that every API was documented and supported.,

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

2.6.3.1 ASPP14:FPT_IDV_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.6 of [ST] states the TOE is versioned with a fixed major and minor version number, as well as a build number that is incremented with every security update to the TOE. The TOE display version information in a major.minor.build format.

Component Guidance Assurance Activities: None Defined

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

The evaluator queried the TOE for its version information using the provided WebUI and verified its existence and ensured it matched the major.minor.build format alongside testing provided for ASPP14:FPT_TUD_EXT.1.2-t1.

2.6.4 Protection of Keys and Key Material (FE10:FPT_KYP_EXT.1)

2.6.4.1 FE10:FPT_KYP_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 the TSS for a high level description of the method(s) used to protect keys stored in non-volatile memory.

KMD

The evaluator shall verify the KMD to ensure it describes the storage location of all keys and the protection of all keys stored in non-volatile memory. The description of the key chain shall be reviewed to ensure FCS_COP.1(5) is followed for the storage of wrapped or encrypted keys in non-volatile memory and plaintext keys in non-volatile memory meet one of the criteria for storage.

Section 6.6 of [ST] states the only key from the keychain specified in FCS_KYC_EXT.1 that is saved to non-volatile memory is the encrypted FEK. The FEK is generated per-session, encrypted with the password-derived Session Key (KEK), and saved in the encrypted file metadata upon completion of the encryption process. Upon decrypting, the decryption utility will rederive the Session key, decrypt the FEK and the file contents, and then write that decrypted data to a new file. The decryption utility does not remove the original file containing the encrypted FEK, however the decryption utility will clear the Session key and decrypted FEK when no longer used.

Outside of the keychain specified in FCS_KYC_EXT.1, the TOE saves its web server credential to the non-volatile storage. This key is never stored in plaintext, but it is always encrypted using a PBKDF-derived key based on a separate, webserver credential passphrase.

The TOE does not claim key wrapping through FCS_COP.1(5) under FPT_KYP_EXT.1, but key encryption through FCS_COP.1(7) and the description of the keychain is consistent with those SFRs .

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

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

2.6.5.1 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 dumped the filesystem of the TOE container and listed all files in the filesystem. The evaluator searched for each library in the list within the vendor-provided SBOM. All libraries were associated with a third-party library consistent with the claims in the ST.

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 3.6 of the AGD describes the process of how updates are performed.

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

The evaluator used application documentation to check for an update. This resulted in a no update available message with no error messages issued.

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 1.2 of the AGD has the procedure for checking the current version of the TOE.

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

The evaluator queried for the application's current version using user guidance. The version matched 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.

The evaluator installed the TOE and collected the md5sum of all executables on the filesystem. The evaluator then performed encryption and decryption on a file. The evaluator then collected the md5sum again and confirmed it is identical to its predecessor.

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 [ST] states that the TOE is a signed, OCI-containerized Linux application, distributed as a part of Shift5's proprietary OS distributed with their Manifold Product line. Updates for the TOE are distributed as a new installation of the TOE container to be installed in place, following the original inner image format. All images are

signed using ECDSA P-384 with SHA-384 using Shift5's private key. The TOE platform will verify the signature on the image using its cryptographic library in conjunction with the vendor's signing key, stored internally, before installing it and will reject any update with an invalid signature.

The TOE can display its current software version and check for updates via the internal WebUI. If a new update is available, the user can download the update through Shift5's product links.

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 [ST] states the TOE can display its current software version and check for updates via the internal WebUI. If a new update is available, the user can download the update through Shift5's product links.

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.

The TOE ships with the proprietary platform OS which does not come with an explicit user command for a factory reset, however the same task can be accomplished by reinstalling the vendor-provided OS software. The evaluator installed the vendor platform in the frontmatter of the Test Report and confirmed that the TOE software was present.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.7 Trusted path/channels (FTP)

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

2.7.1.1 ASPP14:FTP_DIT_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.7 of [ST] states the TOE provides a WebUI to be used by administrators to remotely manage the TOE encryption service. The WebUI itself is implemented by a webserver process within the TOE boundary that binds to a local address and communicates via HTTPS. As a result, the TOE claims compliance for HTTPS as a server and TLS as a server and claims FCS_HTTPS_EXT.1/Server and the Functional Package for TLS accordingly.

The TOE encrypts all transmitted sensitive data. The TOE additionally will request for available update information, however this information is hosted on a publicly accessible web server and is considered non-sensitive as it can be freely accessed by other services.

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 captured packets as part of ASPP14:FDP_NET_EXT.1-t1. The traffic was encrypted with TLS in accordance with the selection in the ST.

Test 2: The evaluator captured packets as part of ASPP14:FDP_NET_EXT.1-t1. No cleartext data was found in the packet capture.

Test 3: The evaluator captured packets as part of ASPP14:FDP_NET_EXT.1-t1. The evaluator performed a string search on the packet capture using wireshark and HxD for any plaintext instances of the WebUI admin account password. The passphrase was not found in either search.

Test 4 – Not applicable as the TOE does not use Android /iOS.

3. Protection Profile SAR Assurance Activities

The following sections address assurance activities specifically defined in the ASPP14/FE10/PKGTLS11 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.

The assurance activities from the PP_APP_V1.4 have been performed and the analysis of the evaluator is documented in the previous sections of this document.

3.2 Guidance documents (AGD)

3.2.1 Operational User Guidance (AGD_OPE.1)

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

evaluation under this PP. The operational guidance shall make it clear to an administrator which security functionality is covered by the evaluation activities.

No user configuration is available for the cryptographic features of the TOE. Section 1.3 of the AGD states that no configuration is needed for the cryptographic modules in order to be compliant. The use of any other cryptographic components beyond those that are identified are neither evaluated nor tested during the TOE's Common Criteria evaluation. The TOE implements compliant TLS protocols which are verified as a part of this evaluation. No additional configuration is needed for the TLS protocol in order to be compliant.

Section 3.6 of the AGD describes how to obtain, install, and verify updates for the TOE. In this section, the AGD states the TOE will automatically verify the digital signature of the update before proceeding.

Section 1.1 of the AGD lists all security functions and management capabilities within the scope of the evaluation.

3.2.2 Preparative Procedures (AGD_PRE.1)

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

Section 2.2 of the AGD identifies all the platforms for the evaluation. The instructions address all the platforms consistent with 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.

The ST identifies the TOE under Section 1.3 as the Shif5, Inc. Software File-Based Encryption Version 1.3. This is consistent with testing and the AGD identification under Section 1.2. The product only has limited public presence as its primarily distributed as a part of the Shift5 Manifold product line, which is sufficiently described as a separate product that has gone through its own, separate evaluation.

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 ALC_CMC.1 for the referenced assurance activities. The Admin Guide identifies a specific version of the TOE for which it is relevant and identifies the platforms for the evaluation. The AGD additionally specifics the supported platforms and prerequisites under Section 2.2. This is consistent with the ST which details out the TOE platform, version identification, operational environments, tested configurations, compilation flags.

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 [ST] states the vendor actively monitors both internal and third-party components and accepts reports of vulnerabilities, bugs, and discrepancies from customers and third parties. Shift5 provides multiple methods for initiating a vulnerability report, including through the assigned Program Management representative during customer onboarding, through the support contacts provided in each customer's support plan (including support@shift5.io), or by submitting a general inquiry through https://shift5.io/contact/ or by calling 703-810-3320 to initiate a support ticket.

The initial contact should contain only sufficient information to establish communication and open a support case. Once the support process has been initiated, sensitive vulnerability information is exchanged using protected communication methods appropriate for the information being shared. Email communications are protected using Transport Layer Security (TLS) while in transit, and Shift5 supports end-to-end encrypted email and file exchange using Virtru when additional protection is required. When requested or required by the customer, Shift5 also supports the use of trusted secure file transfer portals (for example, DoDSAFE) or other approved secure communication channels for transmitting sensitive vulnerability information.

The vendor provides timely security updates for the TOE when vulnerabilities are identified. Shift5 actively monitors both internally developed and third-party software components, including third-party libraries, and incorporates applicable security updates into TOE releases. Reported issues are coordinated through Shift5 engineering, support, and customer-facing teams, triaged, validated, and remediated as appropriate. The reporter is kept informed of the status of the issue throughout the resolution process until the support ticket is closed.

3.4 Tests (ATE)

3.4.1 Independent Testing - Conformance (ATE_IND.1)

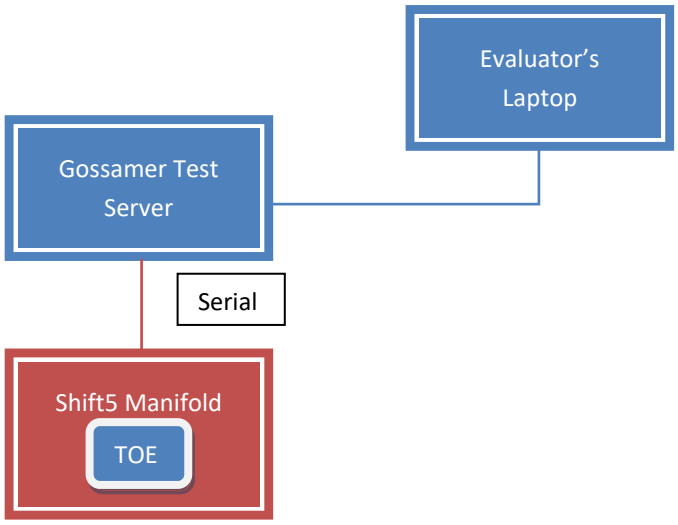
Assurance Activities: The evaluator shall prepare a test plan and report documenting the testing aspects of the system, including any application crashes during testing. The evaluator shall determine the root cause of any application crashes and include that information in the report. The test plan covers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

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

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

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

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



TOE Platforms:

- Shift5 SW FBE Solution – Containerized application installed directly on the Shift5 Manifold Platform, referred to as TOE, Shift5 SW FBE, Shift5 FBE, and DARP (data at rest protection service which is the internal name for the container)

Supporting Products:

- Shift5 Manifold – Hardware-based platform that provides a SUSE Linux Micro 6.0-based Operating System maintained by Shift5
    - Provides an additional layer of Full-disk encryption, evaluated under a separate Common Criteria evaluation and not included in the boundaries of this evaluation

Supporting Software:

- Windows 11 – Evaluator's laptop
    - Windows Security, virus and threat protection
- Windows 10 – Gossamer Test Server
    - Windows 10 Hyper-V Virtualization Platform
    - WinSCP 6.5.6 – Used for file management (SCP protocol)
    - Putty 0.83 – Used for connecting to Ubuntu test server
    - balenaEtcher 2.1.4 – Used for initialing TOE platform OS
    - ZenMap 7.99 – Used for nmap port scans and visualization
    - Wireshark 4.6.4 – Used for analyzing packet captures
- Ubuntu version 24.04 – Gossamer Test Server (VM on Windows 10 machine)
    - tcpdump version 4.99.4 – Used for capturing network traffic
    - Nmap 7.94SVN – Used for capturing network port scans
    - Objdump 2.42 – Used for static analysis
    - Readelf 2.42 – Used for static analysis
    - Hexdump 2.39.3 – Used for static analysis
    - Strings 2.42 – Used for static analysis

3.5 Vulnerability Assessment (AVA)

3.5.1 Vulnerability Survey (AVA_VAN.1)

Assurance Activities: The evaluator shall generate a report to document their findings with respect to this requirement. This report could physically be part of the overall test report mentioned in ATE_IND, or a separate document. The evaluator performs a search of public information to find vulnerabilities that have been found in

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

The evaluator searched the MITRE CVE Database, National Vulnerability Database, and CVE details (https://www.cve.org/, https://web.nvd.nist.gov/vuln/search, and https://www.cvedetails.com/vulnerability-search.php, ref CVE) and Known Vulnerability Exploit Catalog (https://www.cisa.gov/known-exploited-vulnerabilities-catalog, ref KEV) on 7/17/2026 (from 1/1/2020) with the following search terms: "Shift5", "Manifold", "SUSE Linux Micro", "Intel Atom C3708", "Data at rest protection", "Alping ca-certificates", "chainguard", "wolfi", "glibc", "stdlib", "golang", "apk", "apk/libselinux", "apk/lvm2", "apk/pcre2", "apk/popt", "apk/userspace-rcu", "apk/util-linux@", "apk/json-c", "alexedwards/scs/v2", "awnumar/memcall", "awnumar/memguard", "beorn7/perks", "ccoveille/go-safecast", "cespare/xxhash/v2", "fsnotify/fsnotify", "go-logr/logr", "goccy/go-yaml", "golang-migrate/migrate/v4", "google/go-sev-guest", "google/go-tpm-tools", "google/logger", "google/uuid", "hashicorp/errwrap", "hashicorp/go-multierror", "hashicorp/hcl", "justinas/alice", "justinas/nosurf", "lmittmann/tint", "magiconair/properties", "mattn/go-isatty", "mitchellh/mapstructure", "moby/sys/mountinfo", "moby/sys/userns", "munnerz/goautoneg", "pborman/uuid", "pelletier/go-toml/v2", "pkg/errors", "prometheus/client_golang", "prometheus/client_model", "prometheus/common", "prometheus/procfs", "remyoudompheng/bigfft", "rs/cors", "sagikazarmark/slog-shim", "spf13", "spf13/afero", "spf13/cast", "spf13/cobra", "spf13/pflag", "spf13/viper", "subosito/gotenv", "urfave/cli/v3", "veqryn/slog-context", "go.uber.org/atomic", "go.uber.org/automaxprocs", "go.yaml.in/yaml/v2", "golang.org/x/crypto", "golang.org/x/sys", "golang.org/x/text", "google.golang.org/protobuf", "gopkg.in/ini.v1", "gopkg.in/yaml.v3", "k8s.io/klog/v2", "k8s.io/mount-utils", "k8s.io/utils", "modernc.org/libc", "modernc.org/mathutil", "modernc.org/memory", "modernc.org/sqlite", "stdlib", "pk/inih", "openssl", "linux kernel 6.11.3", "util-linux", and “libgcrypt”.

A total of 528 were identified. Of all the sources searched, the following terms identified matches: glibc, openssl, golang, apk, stdlib, golang.org/x/text, golang.org/x/crypto, libgcrypt, util-linux, wolfi. The evaluator found several matches that were not applicable or applicable to an earlier version or patch. For the remaining two issues, the evaluator noted that the TOE contained the referenced library, however the TOE did not include the vulnerable binary and therefore did not contain any exploitable codepath.

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

The evaluator used Windows Security on the TOE's decrypted s5u image and confirmed it contained no viruses.