Shift5, Inc. Software File-Based Encryption Security Target

Version 0.4
07/23/26

Prepared for:

Shift5, Inc.

1100 Wilson Blvd, Ste 2100
Rosslyn, VA 22209

Prepared By:

www.gossamersec.com

LIST OF TABLES

TABLE OF FIGURES

1. Security Target Introduction

This section identifies the Security Target (ST) and Target of Evaluation (TOE) identification, ST conventions, ST conformance claims, and the ST organization. The TOE is Shift5, Inc. Software File-Based Encryption provided by Shift5 Data Systems Corporation. The TOE is being evaluated as a file-based encryption solution.

The Security Target contains the following additional sections:

- Conformance Claims (Section 2)
- Security Objectives (Section 3)
- Extended Components Definition (Section 4)
- Security Requirements (Section 5)
- TOE Summary Specification (Section 6)

Conventions

The following conventions have been applied in this document:

- Security Functional Requirements – Part 2 of the CC defines the approved set of operations that may be applied to functional requirements: iteration, assignment, selection, and refinement.
    - Iteration: allows a component to be used more than once with varying operations. In this ST, iteration may be indicated by a parenthetical number placed at the end of the component. For example FDP_ACC.1(1) and FDP_ACC.1(2) indicate that the ST includes two iterations of the FDP_ACC.1 requirement. Alternately, a usually descriptive textual extension may be added after a slash (/) character to identify a specific iteration. For example, iterations of a requirement such as FCS_COP.1 might be identified as FCS_COP.1/HASH and FCS_COP.1/CRYPT.
    - Assignment: allows the specification of an identified parameter. Assignments are indicated using bold and are surrounded by brackets (e.g., [**assignment**]). Note that an assignment within a selection would be identified in italics and with embedded bold brackets (e.g., [***[selected-assignment]***]).
    - Selection: allows the specification of one or more elements from a list. Selections are indicated using bold italics and are surrounded by brackets (e.g., [***selection***]).
    - Refinement: allows the addition of details. Refinements are indicated using bold, for additions, and strike-through, for deletions (e.g., "… **all** objects …" or "… ~~some~~ **big** things …"). An exception to this marking convention is the case of tables that have cells that content equivalent to nothing (e.g., "None", "No events specified", "NA", "No additional information") where the cell is simply left blank since that represents no meaningful change but is effectively a refinement. Another exception to this marking convention is where extraneous punctuation (e.g., extra brackets) may be omitted or incorrect punctuation (e.g., extra or missing periods) may be corrected, so long as the meaning is not changed.
- Other sections of the ST – Other sections of the ST use bolding to highlight text of special interest, such as captions.

1.1 Security Target Reference

ST Title – Shift5, Inc. Software File-Based Encryption Security Target

ST Version – Version 0.4

ST Date – 07/23/26

1.2 TOE Reference

TOE Identification – Shift5, Inc. Software File-Based Encryption

TOE Developer – Shift5, Inc.

Evaluation Sponsor – Shift5, Inc.

1.3 TOE Overview

The Target of Evaluation (TOE) is Shift5, Inc. Software File-Based Encryption (Shift5 SW FBE) version 1.3.

The TOE provides software File-based Encryption within the computer systems in which it operates.

For the purposes of evaluation, the TOE was evaluated for its compatibility with Shift5's hardware platforms running a proprietary SUSE Micro Linux 6-based operating system. The TOE was fully tested running 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.

1.4 TOE Description

The Shift5 SW FBE is an OCI-containerized software application that Shift5 distributes as a part of their proprietary OS images for their supported hardware platforms (such as their Manifold product line). The TOE provides file-based encryption for designated files and accepts an administratively provided passphrase to enable its file encryption service or decrypt requested files.

The TOE consists of multiple inner processes. The TOE WebUI binds to a local address and provides an HTTPS interface for administrators to login or configure the TOE. The administrator provides the file-encryption passphrase through the command line and starts the file encryption services through the WebUI on the system. Upon unlocking the TOE Data at Rest Protection (DARP) service, the TOE derives the necessary key values, encrypts data using 256-bit AES-XTS, and writes the encrypted results to disk. Lastly, the TOE also has a separate, command-line interface used for decrypting data. The decryption utility takes in the same file-encryption passphrase, validates the password against a saved fingerprint, decrypts the FEK from file metadata, and saves the decrypted file data to a new file.

The Manifold product used for testing also features a separate full-drive encryption layer that is bundled with the TOE Platform, however that is outside the scope of this evaluation and is evaluated separately. Similarly, the TOE features multiple modes of operation, however the TOE requires that only its attended mode of operation is used in order to be compliant with the evaluated configuration. The attended mode of operation is enabled upon default installation of the TOE, which disables all non-attended, or non-password-based modes of authentication.

1.4.1 TOE Architecture

The Product is a locally managed system that provides FBE protection for persistent data stored to an internally monitored directory.

1.4.1.1 Physical Boundaries

The TOE is a containerized software application that is designed to operate inside the Shift5 hardware product lines and interact with the host operating system and additionally installed software. The boundary of the software application is the singular, containerized image pertaining to the FBE solution, distributed by Shift5 as a part of their platform OS.

1.4.1.2 Logical Boundaries

This section summarizes the security functions provided by the Shift5 SW FBE:

- Cryptographic support
- User data protection
- Identification and authentication
- Security management
- Privacy
- Protection of the TSF
- Trusted path/channels

1.4.1.2.1 Cryptographic support

The TOE uses Automated Cryptographic Validation Test System (ACVTS)-validated cryptographic algorithm implementations, provided by two cryptolibraries managed by the TOE developer. All of these cryptographic libraries are installed with the TOE, to support key generation and derivation, encryption/decryption, and establishment of trusted channels to protect data in transit. Using these cryptographic libraries, the TOE implements a file-based encryption scheme to protect sensitive data at rest. The TOE implements a HTTPS/TLS server to securely provide an administrator WebUI used to manage the TOE. The TOE also relies on direct links to hardware entropy provided by Intel CPUs with RDSEED instructions to generate entropy that is used as input data for each of the TOE's deterministic random bit generator (DRBG).

1.4.1.2.2 User data protection

The TOE implements functionality to encrypt sensitive data through its file-based encryption scheme. In order to manage this service, the TOE utilizes command-line level utilities and provides a WebUI interface to enable the file encryption service and manage administrator passphrases. Outside of the network connectivity used by the WebUI and by the user to check for updates, the TOE makes no access to any special hardware or sensitive data repository. The TOE properly protects all data at rest and destroys all references to sensitive and plaintext data upon transitioning to a powered off state.

1.4.1.2.3 Identification and authentication

The TOE provides user authorization based on a file-encryption passphrase in order to protect the keys used for file-based encryption.

1.4.1.2.4 Security management

The TOE requires a file-encryption passphrase, a WebUI access passphrase, a webserver certificate and private key, and a web credential passphrase used to encrypt the webserver private key in storage. The TOE provides default credentials for the WebUI access credentials that must be changed after first login before the WebUI provides any functionality. For all remaining credentials, the TOE does not provide any default values and are configured by the user during the initial provisioning. The TOE provides two management interfaces: a WebUI which can enable/disable the encryption service, and a local console connection which the administrator can use to provide passwords, start the WebUI interface, manage files, and access the separately managed decryption utility. The TOE relies on a new file-encryption passphrase each time the encryption service is started and that same passphrase can be used to later decrypt any files protected during that instance. By default, the TOE is configured with file permissions to protect itself and its data from unauthorized access.

1.4.1.2.5 Privacy

The TOE does not transmit personally identifiable information (PII) over any network interfaces.

1.4.1.2.6 Protection of the TSF

The TOE protects itself against exploitation by implementing address space layout randomization (ASLR) and by not allocating any memory region for both write and execute permission. The TOE uses standard, well-documented APIs and includes a number of third-party libraries used to perform its functions. As a part of the evaluation process, the vendor has provided a SBOM listing all of the integrated 3rd party libraries to NIAP for approval.

The TOE provides the ability to check the version of the TOE as well as to check for TOE updates through its WebUI. TOE software is digitally signed and distributed as a part of the operating system. Updates to the TOE containerized application are signed such that the application platform can cryptographically verify them prior to installation. The TOE does not update its own binary code in any way and when removed, all traces of the TOE application software are deleted.

1.4.1.2.7 Trusted path/channels

The TOE protects communications between itself and a remote administrator using the WebUI interface with TLS/HTTPS.

1.4.2 TOE Documentation

Shift5 File – Based Encryption (FBE) Administrator Guidance Document (AGD), Version 1.3, April 2026 **[Admin Guide]**

2. Conformance Claims

This TOE is conformant to the following CC specifications:

- Common Criteria for Information Technology Security Evaluation Part 2: Security functional components, Version 3.1, Revision 5, April 2017.
    - Part 2 Extended
- Common Criteria for Information Technology Security Evaluation Part 3: Security assurance components, Version 3.1, Revision 5, April 2017.
    - Part 3 Extended
- PP-Configuration for Application Software and File Encryption, Version 1.1, 7 April 2022 (CFG_APP-FE_V1.1)
    - The PP-Configuration includes the following components:
        - Base-PP: Protection Profile for Application Software, Version 1.4 (PP_APP_V1.4)
        - PP-Module: PP-Module for File Encryption, Version 1.0 (MOD_FE_V1.0)
- Package Claims:
    - Functional Package for Transport Layer Security (TLS), Version 1.1, 01 March 2019 (PKGTLS11)

Package	Technical Decision	Applied	Notes
MOD_FE_V1.0	TD0898 - Correction to FDP_PRT_EXT ECD Entry	Yes	Applied
MOD_FE_V1.0	TD0894 - Update Base-PP to PP_APP_V1.4	Yes	Applied
MOD_FE_V1.0	TD0650 - Conformance claim sections updated to allow for MOD_VPNC_V2.3 and 2.4	No	VPNC not claimed
MOD_FE_V1.0	TD0644 - FCS_CKM_EXT.4 test applicability	Yes	Applied
MOD_FE_V1.0	TD0600 - Conformance claim sections updated to allow for MOD_VPNC_V2.3	No	VPNC not claimed
MOD_FE_V1.0	TD0472 - File Encryption SFR Rationale and Consistency of TOE Type added	Yes	Applied
MOD_FE_V1.0	TD0455 - NIST SP800-133 keygen methods for FAK/FEK Generation	Yes	Applied
PKG_TLS_V1.1	TD0779 - Updated Session Resumption Support in TLS package V1.1	Yes	Applied
PKG_TLS_V1.1	TD0770 - TLSS.2 connection with no client cert	No	Not supported
PKG_TLS_V1.1	TD0739 - PKG_TLS_V1.1 has 2 different publication dates	Yes	Applied
PKG_TLS_V1.1	TD0726 - Corrections to (D)TLSS SFRs in TLS 1.1 FP	Yes	Applied
PKG_TLS_V1.1	TD0513 - CA Certificate loading	No	Not supported
PKG_TLS_V1.1	TD0499 - Testing with pinned certificates	No	Not supported
PKG_TLS_V1.1	TD0469 - Modification of test activity for FCS_TLSS_EXT.1.1 test 4.1	Yes	Applied
PKG_TLS_V1.1	TD0442 - Updated TLS Ciphersuites for TLS Package	Yes	Applied
PP_APP_V1.4	TD0964 - Clarifications to FMT_MEC_EXT.1 Windows Test	Yes	Applied
PP_APP_V1.4	TD0945 - Adding FIPS 186-5 in PP_APP_V1.4	Yes	Applied
PP_APP_V1.4	TD0931 - Clarification when CTR_DRBG is Selected for FCS_RBG_EXT.2.2 in PP_APP_V1.4	Yes	Applied
PP_APP_V1.4	TD0914 - Addition of PKG_TLS_V2.0 to Conformance	No	TD marked as optional

Package	Technical Decision	Applied	Notes
	Claims		
PP_APP_V1.4	TD0865 - Consistency of Cryptographic Key Sizes	Yes	Applied
PP_APP_V1.4	TD0844 - Addition of Assurance Package for Flaw Remediation V1.0 Conformance Claim	No	Flaw Remediation not claimed
PP_APP_V1.4	TD0823 - Update to Microsoft Windows Exploit Protection link in FPT_AEX_EXT.1.3	Yes	Applied
PP_APP_V1.4	TD0822 - Correction to Windows Manifest File for FDP_DEC_EXT.1	Yes	Applied
PP_APP_V1.4	TD0815 - Addition of Conditional TSS Activity for FPT_AEX_EXT.1.5	Yes	Applied
PP_APP_V1.4	TD0798 - Static Memory Mapping Exceptions	Yes	Applied
PP_APP_V1.4	TD0780 - FIA_X509_EXT.1 Test 4 Clarification	No	SFR not claimed
PP_APP_V1.4	TD0756 - Update for platform-provided full disk encryption	Yes	Applied
PP_APP_V1.4	TD0747 - Configuration Storage Option for Android	Yes	Applied
PP_APP_V1.4	TD0743 - FTP_DIT_EXT.1.1 Selection exclusivity	Yes	Applied
PP_APP_V1.4	TD0736 - Number of elements for iterations of FCS_HTTPS_EXT.1	Yes	Applied
PP_APP_V1.4	TD0719 - ECD for PP APP V1.3 and 1.4	Yes	Applied
PP_APP_V1.4	TD0717 - Format changes for PP_APP_V1.4	Yes	Applied
PP_APP_V1.4	TD0664 - Testing activity for FPT_TUD_EXT.2.2	No	SFR not claimed
PP_APP_V1.4	TD0650 - Conformance claim sections updated to allow for MOD_VPNC_V2.3 and 2.4	No	VPNC not claimed
PP_APP_V1.4	TD0628 - Addition of Container Image to Package Format	No	SFR not claimed

2.1 Conformance Rationale

The ST conforms to the ASPP14/FE10/PKGTLS11. As explained previously, the security problem definition, security objectives, and security requirements have been drawn from the PP.

3. Security Objectives

The Security Problem Definition may be found in the ASPP14/FE10/PKGTLS11 and this section reproduces only the corresponding Security Objectives for operational environment for reader convenience. The ASPP14/FE10/PKGTLS11 offers additional information about the identified security objectives, but that has not been reproduced here and the ASPP14/FE10/PKGTLS11 should be consulted if there is interest in that material.

In general, the ASPP14/FE10/PKGTLS11 has defined Security Objectives appropriate for software applications / file-based encryption and as such are applicable to the Shift5 SW FBE TOE.

3.1 Security Objectives for the Operational Environment

OE.AUTHORIZATION_FACTOR_STRENGTH An authorized user will be responsible for ensuring that all externally derived authorization factors have sufficient strength and entropy to reflect the sensitivity of the data being protected. This can apply to password or passphrase based, ECC CDH, and RSA authorization factors.

OE.PLATFORM The TOE relies upon a trustworthy computing platform for its execution. This includes the underlying operating system and any discrete execution environment provided to the TOE.

OE.POWER_SAVE The non-mobile operational environment must be configurable so that there exists at least one mechanism that will cause the system to enter a safe power state (A.SHUTDOWN). Any such mechanism (e.g., sleep, hibernate) that does not conform to this requirement must be capable of being disabled. The mobile operational environment must be configurable such that there exists at least one mechanism that will cause the system to lock upon a period of time.

OE.PROPER_ADMIN The administrator of the application software is not careless, willfully negligent or hostile, and administers the software within compliance of the applied enterprise security policy.

OE.PROPER_USER The user of the application software is not willfully negligent or hostile, and uses the software within compliance of the applied enterprise security policy.

OE.STRONG_ENVIRONMENT_CRYPTO The Operating environment will provide a cryptographic function capability that is commensurate with the requirements and capabilities of the TOE.

4. Extended Components Definition

All of the extended requirements in this ST have been drawn from the ASPP14/FE10/PKGTLS11. The ASPP14/FE10/PKGTLS11 defines the following extended requirements and since they are not redefined in this ST the ASPP14/FE10/PKGTLS11 should be consulted for more information in regard to those CC extensions.

Extended SFRs:

- ASPP14:FCS_CKM_EXT.1: Cryptographic Key Generation Services
- FE10:FCS_CKM_EXT.2: File Encryption Key (FEK) Generation
- FE10:FCS_CKM_EXT.3: Key Encrypting Key (KEK) Support
- FE10:FCS_CKM_EXT.4: Cryptographic Key Destruction
- FE10:FCS_CKM_EXT.6: Cryptographic Password/Passphrase Conditioning
- ASPP14:FCS_HTTPS_EXT.1/Server: HTTPS Protocol
- FE10:FCS_IV_EXT.1: Initialization Vector Generation
- FE10:FCS_KDF_EXT.1: Cryptographic Key Derivation Function
- FE10:FCS_KYC_EXT.1: Key Chaining and Key Storage
- ASPP14:FCS_RBG_EXT.1: Random Bit Generation Services
- ASPP14:FCS_RBG_EXT.2: Random Bit Generation from Application
- ASPP14:FCS_STO_EXT.1: Storage of Credentials - per TD0865
- PKGTLS11:FCS_TLS_EXT.1: TLS Protocol
- PKGTLS11:FCS_TLSS_EXT.1: TLS Server Protocol
- FE10:FCS_VAL_EXT.1: Validation
- FE10:FCS_VAL_EXT.2: Validation Remediation
- ASPP14:FDP_DAR_EXT.1: Encryption Of Sensitive Application Data
- ASPP14:FDP_DEC_EXT.1: Access to Platform Resources
- ASPP14:FDP_NET_EXT.1: Network Communications
- FE10:FDP_PM_EXT.1: Protection of Data in Power Managed States
- FE10:FDP_PRT_EXT.1: Protection of Selected User Data
- FE10:FDP_PRT_EXT.2: Destruction of Plaintext Data
- FE10:FIA_AUT_EXT.1: Subject Authorization
- ASPP14:FMT_CFG_EXT.1: Secure by Default Configuration
- ASPP14:FMT_MEC_EXT.1: Supported Configuration Mechanism
- ASPP14:FPR_ANO_EXT.1: User Consent for Transmission of Personally Identifiable
- ASPP14:FPT_AEX_EXT.1: Anti-Exploitation Capabilities
- ASPP14:FPT_API_EXT.1: Use of Supported Services and APIs
- ASPP14:FPT_IDV_EXT.1: Software Identification and Versions
- FE10:FPT_KYP_EXT.1: Protection of Keys and Key Material
- ASPP14:FPT_LIB_EXT.1: Use of Third Party Libraries
- ASPP14:FPT_TUD_EXT.1: Integrity for Installation and Update
- ASPP14:FTP_DIT_EXT.1: Protection of Data in Transit

Extended SARs:

- ALC_TSU_EXT.1: Timely Security Updates

5. Security Requirements

This section defines the Security Functional Requirements (SFRs) and Security Assurance Requirements (SARs) that serve to represent the security functional claims for the Target of Evaluation (TOE) and to scope the evaluation effort.

The SFRs have all been drawn from the ASPP14/FE10/PKGTLS11. The refinements and operations already performed in the ASPP14/FE10/PKGTLS11 are not identified (e.g., highlighted) here, rather the requirements have been copied from the ASPP14/FE10/PKGTLS11 and any residual operations have been completed herein. Of particular note, the ASPP14/FE10/PKGTLS11 made a number of refinements and completed some of the SFR operations defined in the Common Criteria (CC) and that PP should be consulted to identify those changes if necessary.

The SARs are also drawn from the ASPP14/FE10/PKGTLS11. The ASPP14/FE10/PKGTLS11 should be consulted for the assurance activity definitions.

5.1 TOE Security Functional Requirements

The following table identifies the SFRs that are satisfied by Shift5 SW FBE TOE.

Requirement Class	Requirement Component
FCS: Cryptographic support	ASPP14:FCS_CKM.1/AK: Cryptographic Asymmetric Key Generation
	ASPP14:FCS_CKM_EXT.1/PBKDF: Password Conditioning
	ASPP14:FCS_CKM.1/SK: Cryptographic Symmetric Key Generation
	ASPP14:FCS_CKM.2: Cryptographic Key Establishment
	ASPP14:FCS_CKM_EXT.1: Cryptographic Key Generation Services
	FE10:FCS_CKM_EXT.2: File Encryption Key (FEK) Generation
	FE10:FCS_CKM_EXT.3: Key Encrypting Key (KEK) Support
	FE10:FCS_CKM_EXT.4: Cryptographic Key Destruction
	FE10:FCS_CKM_EXT.6: Cryptographic Password/Passphrase Conditioning
	FE10:FCS_COP.1(7): Cryptographic operation (Key Encryption)
	ASPP14:FCS_COP.1/Hash: Cryptographic Operation - Hashing
	ASPP14:FCS_COP.1/KeyedHash: Cryptographic Operation - Keyed-Hash Message Authentication
	ASPP14:FCS_COP.1/Sig: Cryptographic Operation - Signing
	ASPP14:FCS_COP.1/SKC: Cryptographic Operation - Encryption/Decryption
	ASPP14:FCS_HTTPS_EXT.1/Server: HTTPS Protocol
	FE10:FCS_IV_EXT.1: Initialization Vector Generation
	FE10:FCS_KDF_EXT.1: Cryptographic Key Derivation Function
	FE10:FCS_KYC_EXT.1: Key Chaining and Key Storage
	ASPP14:FCS_RBG_EXT.1: Random Bit Generation Services
	ASPP14:FCS_RBG_EXT.2: Random Bit Generation from Application
	ASPP14:FCS_STO_EXT.1: Storage of Credentials
	PKGTLS11:FCS_TLS_EXT.1: TLS Protocol
	PKGTLS11:FCS_TLSS_EXT.1: TLS Server Protocol
	FE10:FCS_VAL_EXT.1: Validation
	FE10:FCS_VAL_EXT.2: Validation Remediation
FDP: User data protection	ASPP14:FDP_DAR_EXT.1: Encryption Of Sensitive Application Data
	ASPP14:FDP_DEC_EXT.1: Access to Platform Resources
	ASPP14:FDP_NET_EXT.1: Network Communications
	FE10:FDP_PM_EXT.1: Protection of Data in Power Managed States
	FE10:FDP_PRT_EXT.1: Protection of Selected User Data

Requirement Class	Requirement Component
	FE10:FDP_PRT_EXT.2: Destruction of Plaintext Data
FIA: Identification and authentication	FE10:FIA_AUT_EXT.1: Subject Authorization
FMT: Security management	ASPP14:FMT_CFG_EXT.1: Secure by Default Configuration
	ASPP14:FMT_MEC_EXT.1: Supported Configuration Mechanism
	ASPP14:FMT_SMF.1: Specification of Management Functions
	FE10:FMT_SMF.1(2): Specification of File Encryption Management Functions
FPR: Privacy	ASPP14:FPR_ANO_EXT.1: User Consent for Transmission of Personally Identifiable
FPT: Protection of the TSF	ASPP14:FPT_AEX_EXT.1: Anti-Exploitation Capabilities
	ASPP14:FPT_API_EXT.1: Use of Supported Services and APIs
	ASPP14:FPT_IDV_EXT.1: Software Identification and Versions
	FE10:FPT_KYP_EXT.1: Protection of Keys and Key Material
	ASPP14:FPT_LIB_EXT.1: Use of Third Party Libraries
	ASPP14:FPT_TUD_EXT.1: Integrity for Installation and Update
FTP: Trusted path/channels	ASPP14:FTP_DIT_EXT.1: Protection of Data in Transit

Table 1 TOE Security Functional Components

5.1.1 Cryptographic support (FCS)

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

ASPP14:FCS_CKM.1.1/AK

The application shall [***implement functionality***] to generate asymmetric cryptographic keys in accordance with a specified cryptographic key generation algorithm [***ECC schemes using 'NIST curves' P-256, P-384 and [P-521] that meet the following: FIPS PUB 186-5, 'Digital Signature Standard (DSS)', Appendix A.2***].
(TD0717 & TD0945 applied)

5.1.1.2 Password Conditioning - per TD0717 & TD0865 (ASPP14:FCS_CKM_EXT.1/PBKDF)

ASPP14:FCS_CKM_EXT.1.1/PBKDF

A password/passphrase shall perform [**PKBDFv2 (HMAC-SHA2-384)**] in accordance with a specified cryptographic algorithm as specified in FCS_COP.1/KeyedHash, with [**310,000**] iterations, and output sizes [***256 bits***] that meet the following: NIST SP 800-132.
(TD0865 applied)

ASPP14:FCS_CKM_EXT.1.2/PBKDF

The TSF shall generate all salts using a RBG that meets FCS_RBG_EXT.1 and with entropy corresponding to the security strength selected for PBKDF in FCS_CKM_EXT.1.1/PBKDF.

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

ASPP14:FCS_CKM.1.1/SK

The application shall generate symmetric cryptographic keys using a Random Bit Generator as specified in FCS_RBG_EXT.1 and specified cryptographic key sizes [***128 bit, 256 bit***].

5.1.1.4 Cryptographic Key Establishment (ASPP14:FCS_CKM.2)

ASPP14:FCS_CKM.2.1

The application shall [***implement functionality***] to perform cryptographic key establishment in accordance with a specified cryptographic key establishment method: [***Elliptic curve-based key establishment schemes that meets the following: NIST Special Publication 800-56A,***

'Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography'].

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

ASPP14:FCS_CKM_EXT.1.1

The application shall [***implement asymmetric key generation***].

5.1.1.6 File Encryption Key (FEK) Generation - per TD0455 (FE10:FCS_CKM_EXT.2)

FE10:FCS_CKM_EXT.2.1

The TSF shall [***generate FEK cryptographic keys [using a Random Bit Generator as specified in FCS_RBG_EXT.1 (from AppPP) and with entropy corresponding to the security strength of AES key sizes of [256 bit],]***]. (TD0455 applied)

FE10:FCS_CKM_EXT.2.2

The TSF shall use a unique FEK for each file (or set of files) using the mechanism on the client as specified in FCS_CKM_EXT.2.1.

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

FE10:FCS_CKM_EXT.3.1

The TSF shall [***generate KEK cryptographic keys [derived from a password/passphrase that is conditioned as defined inFCS_CKM_EXT.6]***].

5.1.1.8 Cryptographic Key Destruction (FE10:FCS_CKM_EXT.4)

FE10:FCS_CKM_EXT.4.1

The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method [

- ***For volatile memory, the destruction shall be executed by a [***
  - ***single overwrite consisting of [zeroes],***
  - ***removal of power to the memory***

],

- ***For non-volatile memory, the destruction shall be executed by [***
  - ***the invocation of an interface provided by the underlying platform that [***
    - ***logically addresses the storage location of the key and performs a [single] overwrite consisting of [a pseudo-random pattern using the TSF's RBG]***

]

]].

FE10:FCS_CKM_EXT.4.2

The TSF shall destroy all keys and key material when no longer needed.

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

FE10:FCS_CKM_EXT.6.1

The TSF shall support a password/passphrase of up to [***128***] characters used to generate a password authorization factor.

FE10:FCS_CKM_EXT.6.2

The TSF shall allow passwords to be composed of any combination of upper case characters, lower case characters, numbers, and the following special characters: '!', '@', '#', '$', '%', '^', '&', '*', '(', and ')', and [[***ASCII printable characters***]].

FE10:FCS_CKM_EXT.6.3

The TSF shall perform Password-based Key Derivation Functions in accordance with a specified cryptographic algorithm HMAC- [***SHA-384***], with [***[310,000] iterations***], and output cryptographic key sizes [***256***] that meet the following: NIST SP 800-132.

FE10:FCS_CKM_EXT.6.4

The TSF shall not accept passwords less than [***[32 characters]***] and greater than the maximum password length defined in FCS_CKM_EXT.6.1.

FE10:FCS_CKM_EXT.6.5

The TSF shall generate all salts using an RBG that meets FCS_RBG_EXT.1 (from AppPP) and with entropy corresponding to the security strength selected for PBKDF in FCS_CKM_EXT.6.3.

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

FE10:FCS_COP.1.1(7)

The TSF shall [***perform key encryption and decryption***] in accordance with a specified cryptographic algorithm AES used in CBC mode and cryptographic key sizes [***256***] bits that meet the following: AES as specified in SP 800-38A.

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

ASPP14:FCS_COP.1.1/Hash

The application shall perform cryptographic hashing services in accordance with a specified cryptographic algorithm [***SHA-256, SHA-384***] and message digest sizes [***256, 384***] bits that meet the following: FIPS Pub 180-4.

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

ASPP14:FCS_COP.1.1/KeyedHash

The application shall perform keyed-hash message authentication in accordance with a specified cryptographic algorithm [***HMAC-SHA-384***] and [***no other algorithms***] with key sizes [**384 bit**] and message digest sizes [***384***] and [***no other size***] bits that meet the following: FIPS Pub 198-1 'The Keyed-Hash Message Authentication Code' and FIPS Pub 180-4 'Secure Hash Standard'. (TD0717 applied)

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

ASPP14:FCS_COP.1.1/Sig

The application shall perform cryptographic signature services (generation and verification) in accordance with a specified cryptographic algorithm [***ECDSA schemes using 'NIST curves' P-256, P-384 and [no other curves] that meet the following: FIPS PUB 186-4 or FIPS 186-5, 'Digital Signature Standard (DSS)', Section 6***].
(TD0717 & TD0945 applied)

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

ASPP14:FCS_COP.1.1/SKC

The application shall perform encryption/decryption in accordance with a specified cryptographic algorithm [

- ***AES-CBC (as defined in NIST SP 800-38A) mode,***
- ***AES-GCM (as defined in NIST SP 800-38D) mode,***
- ***AES-XTS (as defined in NIST SP 800-38E) mode***

] and cryptographic key sizes [***128-bit, 256-bit***[1]].

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

ASPP14:FCS_HTTPS_EXT.1.1/Server

The application shall implement the HTTPS protocol that complies with RFC 2818.

[1] Only AES_GCM supports 128-bit encryption.

ASPP14:FCS_HTTPS_EXT.1.2/Server
The application shall implement HTTPS using TLS as defined in the Functional Package for TLS.
ASPP14:FCS_HTTPS_EXT.1.3/Server
The application shall [***not establish the connection***] if the peer certificate is deemed invalid.
(TD0736 applied)

5.1.1.16 Initialization Vector Generation (FE10:FCS_IV_EXT.1)

FE10:FCS_IV_EXT.1.1
The TSF shall [***generate IVs with the following properties [***
CBC: IVs shall be non-repeating and unpredictable,
XTS: No IV. Tweak values shall be non-negative integers, assigned consecutively, and starting at an arbitrary non-negative integer,
GCM: IV shall be non-repeating. The number of invocations of GCM shall not exceed 2^32 for a given secret key]].

5.1.1.17 Cryptographic Key Derivation Function - per TD0894 (FE10:FCS_KDF_EXT.1)

FE10:FCS_KDF_EXT.1.1
The TSF shall accept [***a conditioned password***] to derive an intermediate key, as defined in [***NIST SP 800-132***] using the keyed-hash functions specified in FCS_COP.1/KeyedHash (from AppPP), such that the output is at least of equivalent security strength (in number of bits) to the FEK.

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

FE10:FCS_KYC_EXT.1.1
The TSF shall maintain a key chain of: [***KEKs originating from [one or more authorization factors(s)] to [the FEK(s)] using the following method(s): [implementation of key encryption as specified in FCS_COP.1(7)] while maintaining an effective strength of [[256 bits] for symmetric keys] commensurate with the strength of the FEK***] and [***no supplemental key chains***].

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

ASPP14:FCS_RBG_EXT.1.1
The application shall [***implement DRBG functionality***] for its cryptographic operations.

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

ASPP14:FCS_RBG_EXT.2.1
The application shall perform all deterministic random bit generation (DRBG) services in accordance with NIST Special Publication 800-90A using [***CTR_DRBG (AES), HMAC_DRBG (any)***].
ASPP14:FCS_RBG_EXT.2.2
The deterministic RBG shall be seeded by an entropy source that accumulates entropy from a platform-based DRBG and [***a hardware-based noise source***] with a minimum of [***256 bits***] of entropy at least equal to the greatest security strength (according to NIST SP 800-57) of the keys and hashes that it will generate.

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

ASPP14:FCS_STO_EXT.1.1
The application shall [***implement functionality to securely store [WebUI Server Certificate] according to [FCS_COP.1/SKC, FCS_CKM_EXT.1/PBKDF]***]
to non-volatile memory.
(TD0865 applied)

5.1.1.22 TLS Protocol (PKGTLS11:FCS_TLS_EXT.1)

PKGTLS11:FCS_TLS_EXT.1.1
The product shall implement [***TLS as a server***]

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

PKGTLS11:FCS_TLSS_EXT.1.1
The product shall implement TLS 1.2 (RFC 5246) and [***no earlier TLS versions***] as a server that supports the cipher suites [
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289,
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289,
] and no other cipher suites, and also supports functionality for [***session resumption based on session tickets according to RFC 5077***], and [***none***] (TD0442 & TD0779 applied, supersedes TD0588)

PKGTLS11:FCS_TLSS_EXT.1.2
The product shall deny connections from clients requesting SSL 2.0, SSL 3.0, TLS 1.0 and [***TLS 1.1***]

PKGTLS11:FCS_TLSS_EXT.1.3
The product shall perform key establishment for TLS using [***ECDHE parameters using elliptic curves [secp256r1, secp384r1, secp521r1] and no other curves***]. (TD0726 applied)

5.1.1.24 Validation - per TD0894 (FE10:FCS_VAL_EXT.1)

FE10:FCS_VAL_EXT.1.1
The TSF shall perform validation of the user by [***validating the [submask] using the following methods: [hash the [submask] as specified in FCS_COP.1/Hash (from AppPP) and compare it to a stored hash]***].

FE10:FCS_VAL_EXT.1.2
The TSF shall require validation of the user prior to decrypting any FEK.

5.1.1.25 Validation Remediation (FE10:FCS_VAL_EXT.2)

FE10:FCS_VAL_EXT.2.1
The TSF shall [***institute a delay such that only [8,640 attempts] can be made within a 24 hour period***].

5.1.2 User data protection (FDP)

5.1.2.1 Encryption Of Sensitive Application Data (ASPP14:FDP_DAR_EXT.1)

ASPP14:FDP_DAR_EXT.1.1
The application shall [***implement functionality to encrypt sensitive data as defined in the PP-module for File Encryption***] in non-volatile memory.

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

ASPP14:FDP_DEC_EXT.1.1
The application shall restrict its access to [***network connectivity***].

ASPP14:FDP_DEC_EXT.1.2
The application shall restrict its access to [***no sensitive information repositories***].

5.1.2.3 Network Communications (ASPP14:FDP_NET_EXT.1)

ASPP14:FDP_NET_EXT.1.1
The application shall restrict network communication to [***user-initiated communication for [checking for updates], respond to [HTTPS WebUI administrator interface]***].

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

FE10:FDP_PM_EXT.1.1
The TSF shall protect all data selected for encryption during the transition to the [***powered-off***] state as per FDP_PRT_EXT.1.1.

FE10:FDP_PM_EXT.1.2
On the return to a powered-on state from the state(s) indicated in FDP_PM_EXT.1.1, the TSF shall authorize the user in the manner specified in FIA_AUT_EXT.1.1 once before any protected data are decrypted.

FE10:FDP_PM_EXT.1.3
The TSF shall destroy all key material and authentication factors stored in plaintext when transitioning to a protected state as defined by FDP_PM_EXT.1.1.

5.1.2.5 Protection of Selected User Data - per TD0894 (FE10:FDP_PRT_EXT.1)

FE10:FDP_PRT_EXT.1.1
The TSF shall perform encryption and decryption of the user-selected file (or set of files) in accordance with FCS_COP.1/SKC (from AppPP).

FE10:FDP_PRT_EXT.1.2
The TSF shall [***implement functionality***] to ensure that all sensitive data created by the TOE when decrypting/encrypting the user-selected file (or set of files) are destroyed in volatile and non-volatile memory when the data is no longer needed according to FCS_CKM_EXT.4.

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

FE10:FDP_PRT_EXT.2.1
The TSF shall [***implement functionality, invoke platform-provided functionality***] to ensure that all original plaintext data created when decrypting/encrypting the user-selected file (or set of files) are destroyed in volatile and non-volatile memory according to FCS_CKM_EXT.4 upon completion of the decryption/encryption operation.

5.1.3 Identification and authentication (FIA)

5.1.3.1 Subject Authorization (FE10:FIA_AUT_EXT.1)

FE10:FIA_AUT_EXT.1.1
The TSF shall [***provide user authorization***] based on [***a password authorization factor conditioned as defined in FCS_CKM_EXT.6***].

5.1.4 Security management (FMT)

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

ASPP14:FMT_CFG_EXT.1.1
The application shall provide only enough functionality to set new credentials when configured with default credentials or no credentials.

ASPP14:FMT_CFG_EXT.1.2
The application shall be configured by default with file permissions which protect the application's binaries and data files from modification by normal unprivileged users.

5.1.4.2 Supported Configuration Mechanism (ASPP14:FMT_MEC_EXT.1)

ASPP14:FMT_MEC_EXT.1.1

The application shall [***invoke the mechanisms recommended by the platform vendor for storing and setting configuration options***].

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

ASPP14:FMT_SMF.1.1

The TSF shall be capable of performing the following management functions [
[Configure WebUI webserver credential password,
Configure file-encryption password authentication factor,
Change WebUI access password authentication factor,
Enable the encryption service,
Decrypt selected files,
Change container log level,
Destroy encrypted file]].

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

FE10:FMT_SMF.1.1(2)

The TSF shall be capable of performing the following management functions: [***change authentication factors, perform a cryptographic erase of the data by the destruction of FEKs or KEKs protecting the FEKs as described in FCS_CKM_EXT.4.1***].

5.1.5 Privacy (FPR)

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

ASPP14:FPR_ANO_EXT.1.1

The application shall [***not transmit PII over a network***].

5.1.6 Protection of the TSF (FPT)

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

ASPP14:FPT_AEX_EXT.1.1

The application shall not request to map memory at an explicit address except for [***no explicit exceptions***].

ASPP14:FPT_AEX_EXT.1.2

The application shall [***not allocate any memory region with both write and execute permissions***].

ASPP14:FPT_AEX_EXT.1.3

The application shall be compatible with security features provided by the platform vendor.

ASPP14:FPT_AEX_EXT.1.4

The application shall not write user-modifiable files to directories that contain executable files unless explicitly directed by the user to do so.

ASPP14:FPT_AEX_EXT.1.5

The application shall be built with stack-based buffer overflow protection enabled.

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

ASPP14:FPT_API_EXT.1.1

The application shall use only documented platform APIs.

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

ASPP14:FPT_IDV_EXT.1.1
The application shall be versioned with [***[major and minor version and build number]***].

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

FE10:FPT_KYP_EXT.1.1
The TSF shall [***store keys in non-volatile memory only when [encrypted, as specified in FCS_COP.1(7), the plaintext key is not part of the key chain as specified in FCS_KYC_EXT.1]***].

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

ASPP14:FPT_LIB_EXT.1.1
The application shall be packaged with only [***libraries identified in vendor-provided SBOM***].

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

ASPP14:FPT_TUD_EXT.1.1
The application shall [***provide the ability***] to check for updates and patches to the application software.
ASPP14:FPT_TUD_EXT.1.2
The application shall [***provide the ability***] to query the current version of the application software.
ASPP14:FPT_TUD_EXT.1.3
The application shall not download, modify, replace or update its own binary code.
ASPP14:FPT_TUD_EXT.1.4
Application updates shall be digitally signed such that the application platform can cryptographically verify them prior to installation.
ASPP14:FPT_TUD_EXT.1.5
The application is distributed [***with the platform OS***].

5.1.7 Trusted path/channels (FTP)

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

ASPP14:FTP_DIT_EXT.1.1
The application shall [***encrypt all transmitted [sensitive data] with [HTTPS as a server in accordance with FCS_HTTPS_EXT.1/Server for [Administrator WebUI], TLS as a server as defined in the Functional Package for TLS and also supports functionality for [none] for [Administrator WebUI]]***]
between itself and another trusted IT product.
(TD0743 applied)

5.2 TOE Security Assurance Requirements

The SARs for the TOE are the components as specified in Part 3 of the Common Criteria. Note that the SARs have effectively been refined with the assurance activities explicitly defined in association with both the SFRs and SARs.

Requirement Class	Requirement Component
ADV: Development	ADV_FSP.1: Basic Functional Specification
AGD: Guidance documents	AGD_OPE.1: Operational User Guidance
	AGD_PRE.1: Preparative Procedures
ALC: Life-cycle support	ALC_CMC.1: Labelling of the TOE
	ALC_CMS.1: TOE CM Coverage

	ALC_TSU_EXT.1: Timely Security Updates
ATE: Tests	ATE_IND.1: Independent Testing - Conformance
AVA: Vulnerability assessment	AVA_VAN.1: Vulnerability Survey

Table 2 Assurance Components

5.2.1 Development (ADV)

5.2.1.1 Basic Functional Specification (ADV_FSP.1)

ADV_FSP.1.1d
The developer shall provide a functional specification.

ADV_FSP.1.2d
The developer shall provide a tracing from the functional specification to the SFRs.

ADV_FSP.1.1c
The functional specification shall describe the purpose and method of use for each SFR-enforcing and SFR-supporting TSFI.

ADV_FSP.1.2c
The functional specification shall identify all parameters associated with each SFR-enforcing and SFR-supporting TSFI.

ADV_FSP.1.3c
The functional specification shall provide rationale for the implicit categorization of interfaces as SFR-non-interfering.

ADV_FSP.1.4c
The tracing shall demonstrate that the SFRs trace to TSFIs in the functional specification.

ADV_FSP.1.1e
The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_FSP.1.2e
The evaluator shall determine that the functional specification is an accurate and complete instantiation of the SFRs.

5.2.2 Guidance documents (AGD)

5.2.2.1 Operational User Guidance (AGD_OPE.1)

AGD_OPE.1.1d
The developer shall provide operational user guidance.

AGD_OPE.1.1c
The operational user guidance shall describe, for each user role, the user-accessible functions and privileges that should be controlled in a secure processing environment, including appropriate warnings.

AGD_OPE.1.2c
The operational user guidance shall describe, for each user role, how to use the available interfaces provided by the TOE in a secure manner.

AGD_OPE.1.3c
The operational user guidance shall describe, for each user role, the available functions and interfaces, in particular all security parameters under the control of the user, indicating secure values as appropriate.

AGD_OPE.1.4c
The operational user guidance shall, for each user role, clearly present each type of security-relevant event relative to the user-accessible functions that need to be performed, including changing the security characteristics of entities under the control of the TSF.

AGD_OPE.1.5c
The operational user guidance shall identify all possible modes of operation of the TOE (including operation following failure or operational error), their consequences, and implications for maintaining secure operation.

AGD_OPE.1.6c
The operational user guidance shall, for each user role, describe the security measures to be followed in order to fulfill the security objectives for the operational environment as described in the ST.

AGD_OPE.1.7c
The operational user guidance shall be clear and reasonable.

AGD_OPE.1.1e
The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.2.2 Preparative Procedures (AGD_PRE.1)

AGD_PRE.1.1d
The developer shall provide the TOE, including its preparative procedures.

AGD_PRE.1.1c
The preparative procedures shall describe all the steps necessary for secure acceptance of the delivered TOE in accordance with the developer's delivery procedures.

AGD_PRE.1.2c
The preparative procedures shall describe all the steps necessary for secure installation of the TOE and for the secure preparation of the operational environment in accordance with the security objectives for the operational environment as described in the ST.

AGD_PRE.1.1e
The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AGD_PRE.1.2e
The evaluator shall apply the preparative procedures to confirm that the TOE can be prepared securely for operation.

5.2.3 Life-cycle support (ALC)

5.2.3.1 Labelling of the TOE (ALC_CMC.1)

ALC_CMC.1.1d
The developer shall provide the TOE and a reference for the TOE.

ALC_CMC.1.1c
The application shall be labelled with a unique reference.

ALC_CMC.1.1e
The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.3.2 TOE CM Coverage (ALC_CMS.1)

ALC_CMS.1.1d
The developer shall provide a description in the TSS of how timely security updates are made to the TOE. Application developers must support updates to their products for purposes of fixing security vulnerabilities.

ALC_CMS.1.1c
The configuration list shall include the following: the TOE itself; and the evaluation evidence required by the SARs.

ALC_CMS.1.2c
The configuration list shall uniquely identify the configuration items.

ALC_CMS.1.1e
The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.3.3 Timely Security Updates (ALC_TSU_EXT.1)

ALC_TSU_EXT.1.1d
The developer shall provide a description in the TSS of how timely security updates are made to the TOE. Note: Application developers must support updates to their products for purposes of fixing security vulnerabilities.

ALC_TSU_EXT.1.2d
The developer shall provide a description in the TSS of how users are notified when updates change security properties or the configuration of the product.

ALC_TSU_EXT.1.1c
The description shall include the process for creating and deploying security updates for the TOE software.

ALC_TSU_EXT.1.2c
The description shall express the time window as the length of time, in days, between public disclosure of a vulnerability and the public availability of security updates to the TOE.

ALC_TSU_EXT.1.3c
The description shall include the mechanisms publicly available for reporting security issues pertaining to the TOE.

Note: The reporting mechanism could include web sites, email addresses, as well as a means to protect the sensitive nature of the report (e.g., public keys that could be used to encrypt the details of a proof-of-concept exploit).

ALC_TSU_EXT.1.1e
The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.4 Tests (ATE)

5.2.4.1 Independent Testing - Conformance (ATE_IND.1)

ATE_IND.1.1d
The developer shall provide the TOE for testing.

ATE_IND.1.1c
The TOE shall be suitable for testing.

ATE_IND.1.1e
The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ATE_IND.1.2e
The evaluator shall test a subset of the TSF to confirm that the TSF operates as specified.

5.2.5 Vulnerability assessment (AVA)

5.2.5.1 Vulnerability Survey (AVA_VAN.1)

AVA_VAN.1.1d
The developer shall provide the TOE for testing.

AVA_VAN.1.1c
The TOE shall be suitable for testing.

AVA_VAN.1.1e
The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_VAN.1.2e

The evaluator shall perform a search of public domain sources to identify potential vulnerabilities in the TOE.

AVA_VAN.1.3e

The evaluator shall conduct penetration testing, based on the identified potential vulnerabilities, to determine that the TOE is resistant to attacks performed by an attacker possessing Basic attack potential.

6. TOE Summary Specification

This chapter describes the security functions:

- Cryptographic support
- User data protection
- Identification and authentication
- Security management
- Privacy
- Protection of the TSF
- Trusted path/channels

6.1 Cryptographic support

ASPP14:FCS_CKM.1/AK,

ASPP14:FCS_CKM_EXT.1:

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.

ASPP14:FCS_CKM_EXT**.1/PBKDF:**

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.

ASPP14:FCS_CKM.1/SK,

FE10:FCS_CKM_EXT.2:

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.

ASPP14:FCS_CKM.2:

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.

FE10:FCS_CKM_EXT.3:

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.

FE10:FCS_CKM_EXT.4:

The TOE stores plaintext keys in both volatile and non-volatile memory.

The TOE stores the following file encryption keys in volatile memory:

- The administrator file-encryption passphrase (variable length) – The TOE DARP service uses this value to validate the user and derive the Session Key used to encrypt FEK values. Upon startup of the WebUI, the demonstrator provides the password via command line interface. The administrator then logs on to the WebUI to start the encryption process and process the password to produce the file encryption keys. Once the KEK is derived, the TOE clears this sensitive value from memory. Similarly, the administrator provides the file encryption password via command line interaction with the decryption utility. Within the decryption process, the decryption utility will clear this value from memory upon derivation of the Session Key.
- The Session Key (KEK, AES 256b) – The TOE DARP service uses this value to encrypt FEKs before writing them to disk. This value is kept in memory until the TOE encryption service is disabled or the TOE shuts down. Upon disabling the TOE encryption service, the KEK is cleared from memory. Similarly, within the decryption process, the decryption utility will clear this value from memory upon decryption of the FEK.
- The File Encryption Key (FEK, AES 256b) - The TOE DARP service generates a FEK for each individual session. Upon completion of file encryption, the TOE will encrypt the FEK using the KEK and write the encrypted value to the file metadata. Upon shutting down the encryption service, the TOE clears the FEK from memory. Similarly, within the decryption process, the decryption utility will clear this value from memory upon decryption of the file data.
- Plaintext data – Upon completion of encrypting file data, the TOE DARP service clears all plaintext data buffers from memory. Similarly, within the decryption process, the decryption utility will clear this value from memory upon finishing decryption of the file data.

The TOE stores the following keys in non-volatile memory:

- Encrypted FEK – 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 support for cryptographic erase of data by destruction of FEKs or KEKs as a part of FE10:FMT_SMF.1.1(2), the TOE documents a method of using the platform shred utility that will logically overwrite the logical representation of memory using a pseudo-random pattern prior to a secure delete. This shredding process destroys the encrypted data, the encrypted FEK, the PBKDF salt needed to rederive the KEK, and the password validation fingerprint. In cases where the TOE is rebooted while the file encryption service is running, the TOE relies on removal of power to clear key values. For all other instances where the TOE is destroying sensitive data, the TOE will overwrite the values with a single-overwrite of zeros.

In cases where the TOE is rebooted while the file encryption service is running, the TOE relies on removal of power to clear key values. For all other instances where the TOE is destroying sensitive data, the TOE will overwrite the values with a single-overwrite of zeros.

To mitigate physical persistence of data, the underlying drive and platform (including any potential RAID array) must support the TRIM command, support TRIM over its connection channel (e.g., PCI-Express), and implement garbage collection. The drive must be healthy and retired before significant damage occurs. The Shift5 FBE application does everything in its power to sanitize and destroy its sensitive values when no longer needed, however end users should be aware of potential platform operations outside the TOE boundary that could interfere with memory clearing, such as platform clipboard and shell history.

FE10:FCS_CKM_EXT.6:

The TOE utilizes three separate passphrases: a passphrase to access the WebUI, a file-encryption passphrase (provided to DARP service through the WebUI), and a passphrase to encrypt/decrypt the webserver private key. 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 service to enable the encryption service. Additionally, the WebUI web server private key is stored in a password-based, encrypted manner. This webserver private key 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 the file-encryption passphrase is the only passphrase tied to the file encryption key hierarchy (as per FE10:FCS_CKM_EXT.2, FE10:FCS_CKM_EXT.3, FE10:FIA_AUT_EXT.1), the file-encryption passphrase meets the requirements for this SFR.

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.

FE10:FCS_COP.1(7),

ASPP14:FCS_COP.1/Hash,

ASPP14:FCS_COP.1/KeyedHash,

ASPP14:FCS_COP.1/Sig,

ASPP14:FCS_COP.1/SKC:

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 3 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 AES-128/256 GCM Encrypt / Decrypt	NIST SP 800-38A NIST SP 800-38D	A7329
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 4 Shift5 OpenSSL-based Cryptographic Library Cryptographic Algorithms

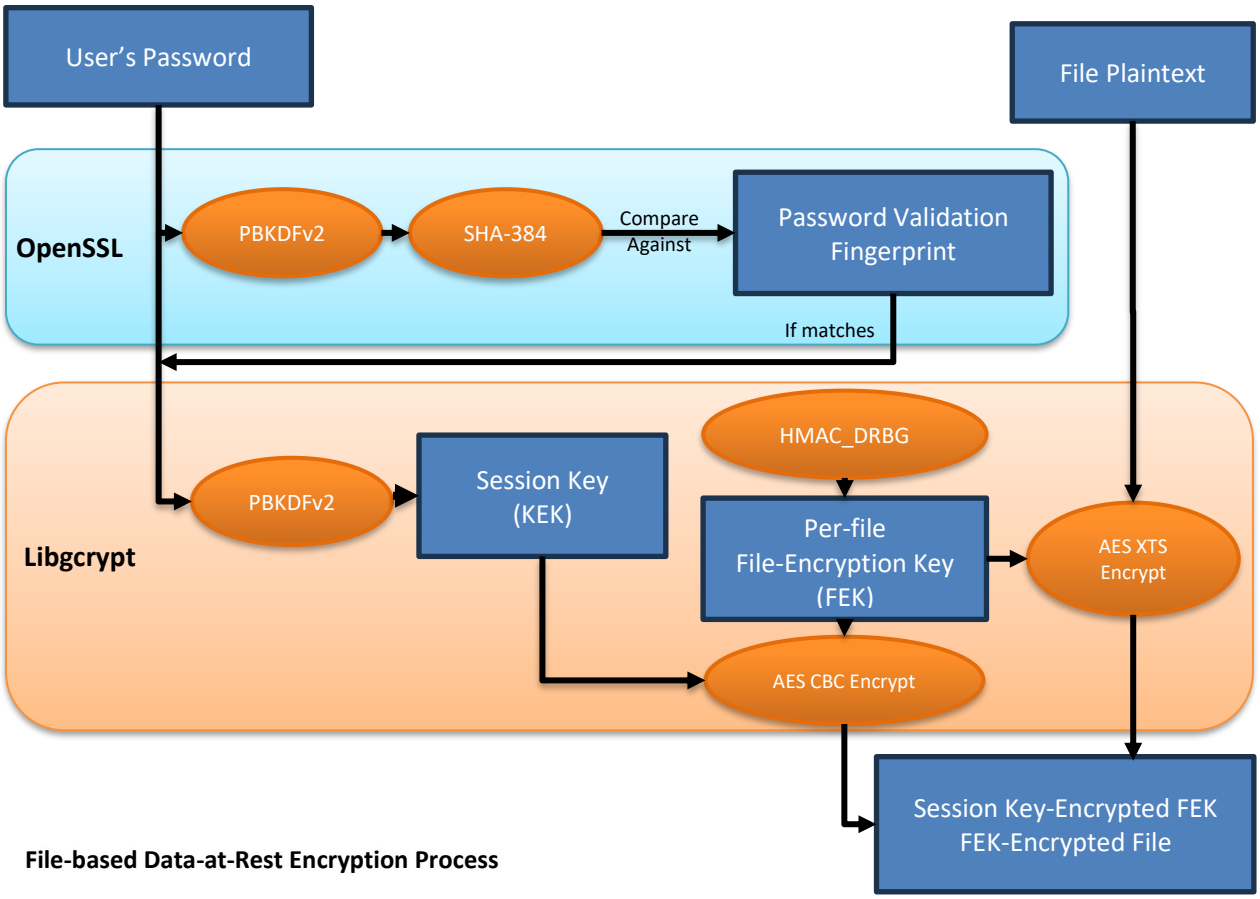

Figure 1: Shift5 File-based Data-at-Rest Encryption Diagram

Key Strength:

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.

Key Encryption:

The TOE relies on key encryption for securing FEK values in file metadata by encrypting them with AES-256 in CBC mode using the Session Key value derived from the administrator file-encryption passphrase as a KEK.

Hashing:

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.

Keyed Hashing:

The TOE's Libgcrypt library implements HMAC-SHA-384 using 256-bit keys, the SHA-384 hash algorithm, a 1024-bit block size, and an output MAC length of 384 bits for the respective passphrase verification and KEK derivation actions within the DARP service. The TOE uses HMAC-SHA-384 as part of all PBKDF (NIST SP 800-132) key management operations.

Signature Generation / Verification:

The TOE generates a digital signature while acting as a TLS server, as part of the TLS certificate message.

Symmetric Encryption / Decryption:

The TOE's Libgcrypt library uses an AES XTS cryptographic implementation dedicated to drive encryption/decryption of protected files. This implementation uses AES 256-bit keys.

The TOE's OpenSSL library also uses AES-CBC to encrypt the WebUI's webserver private key. Upon startup, the administrator will provide a passphrase value to decrypt the webserver's private key and start the internal TLS proxy. The passphrase will be conditioned into an AES 256 key and used to AES-CBC decrypt the server credentials. The TOE negotiates AES-GCM 128 and 256-bit ciphers as a part of TLS connection.

ASPP14:FCS_HTTPS_EXT.1/Server:

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.

ASPP14:FCS_IV_EXT.1:

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.

ASPP14:FCS_KDF_EXT.1:

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.

ASPP14:FCS_KYC_EXT.1:

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.

ASPP14:FCS_RBG_EXT.1,

ASPP14:FCS_RBG_EXT.2:

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.

For the TOE's OpenSSL-based cryptographic library, the TOE uses the DRBG to generate a random 16-byte salt value to append to the raw file-encryption passphrase and the passphrase used to encrypt the webserver certificate before feeding it through the PBKDF passphrase conditioning function. The resulting FBE password-derived value is hashed and compared to a saved copy for passphrase validation before continuing with any attempt to encrypt files. The resulting webserver certificate password-derived value is used as an AES CBC encryption key that is used to decrypt the webserver credentials at startup of the WebUI webserver. Additionally, the OpenSSL used by the WebUI also relies on the RBG for all cryptographic random used in the TLS handshake communication.

For the TOE's libgrcypt-based cryptographic library, the TOE similarly generates a random 16-byte salt value, appends this value to the raw file-encryption passphrase, and then feeds the result through the PBKDF passphrase conditioning function. The resulting value is used a Session key (KEK) which protects the FEK. The libgcrypt-based cryptographic library then will also generate a new 256-bit AES-XTS key using the cryptolibrary's internal DRBG. All FEKs are stored in the file-meta data after being encrypted by the Session Key.

The TOE instructs users to overwrite data with pseudorandom data in order destroy any encrypted, sensitive data when no longer needed. In order to do this, the user simply uses the platform dd utility to copy from the /dev/urandom kernel device file. This pulls data from the same platform resource used to seed the TOE's cryptographic libraries that originate from the platform's hardware noise source.

ASPP14:FCS_STO_EXT.1:

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.

PKGTLS11:FCS_TLS_EXT.1,

PKGTLS11:FCS_TLSS_EXT.1:

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.

FE10:FCS_VAL_EXT.1,

FE10:FCS_VAL_EXT.2:

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.

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.

6.2 User data protection

ASPP14:FDP_DAR_EXT.1:

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.

ASPP14:FDP_DEC_EXT.1:

The TOE does not make any claims for special access to hardware resources or sensitive information repositories other than its need for network connectivity to provide a WebUI for administrator configuration.

ASPP14:FDP_NET_EXT.1:

The TOE restricts its network connection to respond to requests to its HTTPS WebUI Interface and requests by the user to check for version updates. The TOE primarily hosts the web UI under TCP port 32713 intended for client access. The TOE also opens a secondary external port within the standard NodePort range (30000-32767) intended for container compatibility. Both external ports map to the exact same internal container port of 43016 and provide access to the same WebUI HTTPS service.

FE10:FDP_PM_EXT.1:

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.**FE10:FDP_PRT_EXT.1,**

FE10:FDP_PRT_EXT.2:

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.

6.3 Identification and authentication

FE10:FIA_AUT_EXT.1:

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.

6.4 Security management

ASPP14:FMT_CFG_EXT.1:

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, 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.

ASPP14:FMT_MEC_EXT.1:

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 platform'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.

ASPP14:FMT_SMF.1,

FE10:FMT_SMF.1(2):

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.

6.5 Privacy

ASPP14:FPR_ANO_EXT.1:

The TOE does not transmit any information over a network.

6.6 Protection of the TSF

ASPP14:FPT_AEX_EXT.1:

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.

ASPP14:FPT_API_EXT.1:

The TOE uses the following list of documented APIs:

- github.com/alexedwards/scs/v2
- github.com/awnumar/memguard
- github.com/fsnotify/fsnotify
- github.com/golang-migrate/migrate/v4
- github.com/golang-migrate/migrate/v4/database/sqlite

- github.com/golang-migrate/migrate/v4/source/iofs
- github.com/google/uuid
- github.com/justinas/alice
- github.com/justinas/nosurf
- github.com/lmittmann/tint
- github.com/rs/cors
- github.com/shift5-software-engineering/cgo-cryptsetup
- github.com/shift5-software-engineering/cgo-manifold-darp-service/internal/block
- github.com/shift5-software-engineering/cgo-manifold-darp-service/internal/config
- github.com/shift5-software-engineering/cgo-manifold-darp-service/internal/crypto
- github.com/shift5-software-engineering/cgo-manifold-darp-service/internal/dto
- github.com/shift5-software-engineering/cgo-manifold-darp-service/internal/logging
- github.com/shift5-software-engineering/cgo-manifold-darp-service/internal/react
- github.com/shift5-software-engineering/cgo-manifold-darp-service/internal/repo
- github.com/shift5-software-engineering/cgo-manifold-darp-service/internal/secure
- github.com/shift5-software-engineering/cgo-manifold-darp-service/internal/server
- github.com/shift5-software-engineering/cgo-manifold-darp-service/internal/server/endpoints/api/v1
- github.com/shift5-software-engineering/cgo-manifold-darp-service/internal/server/endpoints/endpointutils
- github.com/shift5-software-engineering/cgo-manifold-darp-service/internal/server/endpoints/utils
- github.com/shift5-software-engineering/cgo-manifold-darp-service/internal/server/middleware
- github.com/shift5-software-engineering/cgo-manifold-darp-service/internal/service
- github.com/shift5-software-engineering/cgo-manifold-darp-service/internal/user
- github.com/shift5-software-engineering/cgo-manifold-darp-service/pkg/dispatcher
- github.com/shift5-software-engineering/cgo-manifold-darp-service/pkg/filewatcher
- github.com/shift5-software-engineering/cgo-manifold-darp-service/pkg/task
- github.com/shift5-software-engineering/gcrypt
- github.com/spf13/cobra
- github.com/spf13/viper
- github.com/veqryn/slog-context
- golang braces.dev/errtrace
- golang bytes
- golang context
- golang crypto/fips140
- golang crypto/pbkdf2
- golang crypto/rand
- golang crypto/sha256
- golang crypto/sha512
- golang crypto/subtle
- golang database/sql
- golang embed
- golang encoding/gob
- golang encoding/hex
- golang encoding/json
- golang errors
- golang flag
- golang fmt
- golang go.uber.org/automaxprocs
- golang golang.org/x/exp/slog
- golang golang.org/x/sync/singleflight
- golang golang.org/x/sys/unix
- golang gopkg.in/yaml.v2
- golang io
- golang io/fs

- golang k8s.io/mount-utils
- golang log
- golang log/slog
- golang maps
- golang math
- golang modernc.org/sqlite
- golang net/http
- golang net/http/httputil
- golang net/url
- golang os
- golang os/exec
- golang os/signal
- golang path
- golang path/filepath
- golang regexp
- golang runtime
- golang runtime/debug
- golang slices
- golang strconv
- golang strings
- golang sync
- golang syscall
- golang testing
- golang time
- golang unsafe

ASPP14:FPT_IDV_EXT.1:

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.

FE10:FPT_KYP_EXT.1:

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.

ASPP14:FPT_LIB_EXT.1:

The TOE is designed to run on the Shift5 Manifold platform, a SUSE Linux Micro 6.0 system and utilizes several 3rd party libraries as identified in the vendor-provided SBOM provided to NIAP as a part of this evaluation.

ASPP14:FPT_TUD_EXT.1,

ASPP14:ALC_CMS.1,

ASPP14:ALC_TSC_EXT.1:

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.

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.

6.7 Trusted path/channels

ASPP14:FTP_DIT_EXT.1:

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.