Red Hat, Inc.

Red Hat Enterprise Linux 9.0 EUS

Assurance Activity Report

Version 0.4

January 2024

Document prepared by

www.lightshipsec.com

Document History

Version	Date	Author	Reviewer	Description
0.1	27-Feb-2023	G. McLearn		Initial ASE AAR verdicts.
0.2	12/14/2023	K. Yoshino	C. Cantlon	Completed evaluation activities.
0.3	1/4/2024	K. Yoshino	K. Steiner	Addressed ECR comments.
0.4	1/10/2024	K. Yoshino		Minor AVA Updates.

Table of Contents

1 Introduction

1 This Assurance Activity Report (AAR) documents the evaluation activities performed by Lightship Security for the evaluation identified in Table 1. The AAR is produced in accordance with National Information Assurance Program (NIAP) reporting guidelines.

1.1 Evaluation Identifiers

Table 1: Evaluation Identifiers

Scheme	NIAP Common Criteria Evaluation and Validation Scheme
Evaluation Facility	Lightship Security USA
Developer/Sponsor	Red Hat, Inc.
TOE	Red Hat Enterprise Linux 9.0 EUS
Security Target	Red Hat Enterprise Linux 9.0 EUS Security Target, v1.1
Protection Profile	Protection Profile for General Purpose Operating Systems, Version 4.3, 2022-09-27 (PP_OS_V4.3) Functional Package for Transport Layer Security (TLS), Version 1.1, 2019-03-01 (PKG_TLS_V1.1) Functional Package for Secure Shell (SSH), Version 1.0, 2021-05-13 (PKG_SSH_V1.0)

1.2 Evaluation Methods

2 The evaluation was performed using the methods and standards identified in Table 2.

Table 2: Evaluation Methods

Evaluation Criteria	CC v3.1R5
Evaluation Methodology	CEM v3.1R5
Supporting Documents	Protection Profile for General Purpose Operating Systems, Version 4.3, 2022-09-27 (PP_OS_V4.3) Functional Package for Transport Layer Security (TLS), Version 1.1, 2019-03-01 (PKG_TLS_V1.1) Functional Package for Secure Shell (SSH), Version 1.0, 2021-05-13 (PKG_SSH_V1.0)
Technical Decisions	See Table 3

Table 3: Technical Decisions

TD #	Name	Source	Applicability Rationale
TD0442	Updated TLS Ciphersuites for TLS Package	PKG_TLS_V1.1	Applicable
TD0469	Modification of test activity for FCS_TLSS_EXT.1.1 test 4.1	PKG_TLS_V1.1	Not Applicable - FCS_TLSS_EXT.1 not claimed.
TD0499	Testing with pinned certificates	PKG_TLS_V1.1	Applicable
TD0513	CA Certificate loading	PKG_TLS_V1.1	Applicable
TD0675	Make FPT_W^X_EXT.1 Optional	PP_OS_V4.3	Applicable
TD0682	Addressing Ambiguity in FCS_SSHS_EXT.1 Tests	PKG_SSH_V1.0	Applicable
TD0691	OSPP 4.3 Conditional authentication testing	PP_OS_V4.3	Applicable
TD0693	Typos in OSPP 4.3	PP_OS_V4.3	Applicable
TD0695	Choice of 128 or 256 bit size in AES-CTR in SSH Functional Package.	PKG_SSH_V1.0	Applicable
TD0696	Removal of 160 bit selection from FCS_COP.1/HASH & FCS_COP.1/KEYHMAC	PP_OS_V4.3	Applicable
TD0701	Incomplete selection references in FCS_CKM_EXT.4 TSS activities	PP_OS_V4.3	Applicable
TD0712	Support for Bluetooth Standard 5.3	PP_OS_V4.3	Applicable
TD0713	Functional Package SFR mappings to objectives	PP_OS_V4.3	Applicable
TD0726	Corrections to (D)TLSS SFRs in TLS 1.1 FP	PKG_TLS_V1.1	Not Applicable - FCS_TLSS_EXT.1 not claimed.
TD0732	FCS_SSHS_EXT.1.3 Test 2 Update	PKG_SSH_V1.0	Applicable
TD0739	PKG_TLS_V1.1 has 2 different publication dates	PKG_TLS_V1.1	Applicable
TD0770	TLSS.2 connection with no client cert	PKG_TLS_V1.1	Not Applicable – FCS_TLSS_EXT.1 not claimed.
TD0773	Updates to FIA_X509_EXT.1 for Exception Processing and Test Conditions	PP_OS_V4.3	Applicable
TD0777	Clarification to Selections for Auditable Events for FCS_SSH_EXT.1	PKG_SSH_V1.0	Applicable

TD #	Name	Source	Applicability Rationale
TD0779	Updated Session Resumption Support in TLS package V1.1	PKG_TLS_V1.1	Not Applicable – FCS_TLSS_EXT.1 not claimed.
TD0789	Correction to TLS Selection in FIA_X509_EXT.2.1	PP_OS_V4.3	Applicable
TD0809	Update to FCS_COP.1/SIGN for CNSA 1.0 compliance with Secure Boot exception	PP_OS_V4.3	Applicable
TD0812	Updated CC Conformance Claims in PP_OS_V4.3	PP_OS_V4.3	Applicable

1.3 Reference Documents

Table 4: List of Reference Documents

Ref	Document
[ST]	Red Hat Enterprise Linux 9.0 EUS Security Target, v1.1
[AGD]	Red Hat Enterprise Linux 9.0 EUS Common Criteria Guide, Version 1.1
[RHEL]	Red Hat Enterprise Linux 9, Configuring basic system settings, 2023-11-08

2 Test Overview

3 Testing was performed by Nhien Truong, Kenji Yoshino, and Nil Folquer from May 2023 through January 2024.

4 Testing of the Dell platform was performed in the Lightship Baltimore facility that has been accredited by NVLAP. The TOE on Dell platform and test setup was physically and logically protected from unauthorized access, so the integrity of the TOE and test results can be assured.

5 Testing of the IBM z16 and Power10 platforms was performed remotely, with both systems secured in IBM datacenters. Remote testing was approved by NIAP on June 7, 2023. All communication with the TOE for executing tests and collecting evidence was performed over AES-256 encrypted VPN channels.

2.1 TOE Components

6 The TOE is a software TOE and is comprised of the following:

- Red Hat Enterprise Linux 9.0 Extended Update Support

7 The TOE is downloaded by users at: https://access.redhat.com/

8 The TOE does not have any physical components. The TOE runs on one of the compute platforms specified in section 2.3.

2.2 Version Verification

9 The provided TOE is labelled with its reference as found in the ST. The TOE displayed “Red Hat Enterprise Linux release 9.0 (Plow)”. The kernel version is kernel-5.14.0-70.70.1.el9_0, “el9_0” indicating “EUS.”

2.3 Non-TOE Components

10 The following components must be present in the operational environment to operate the TOE in the evaluated configuration:

- **Update Server.** The TOE receives updates from an organization’s local repository via TLS.
- **SSH Server.** The TOE is capable of securely communicating with an SSHv2 server.
- **SSH Client.** The TOE is capable of securely communicating with an SSHv2 client.
- **Compute Platform.** The TOE requires one of the following compute platforms meeting the following specifications:
  - Intel Xeon Silver x86-64 UEFI platforms (of Cascade Lake microarchitecture)
  - IBM z16 (LPAR) platforms with UEFI 2.3.1 or later.
  - Power10 PowerVM (LPAR) platforms with UEFI 2.3.1 or later.

2.4 Test Environment

11 Figure 1 shows a logical view of the test setup.

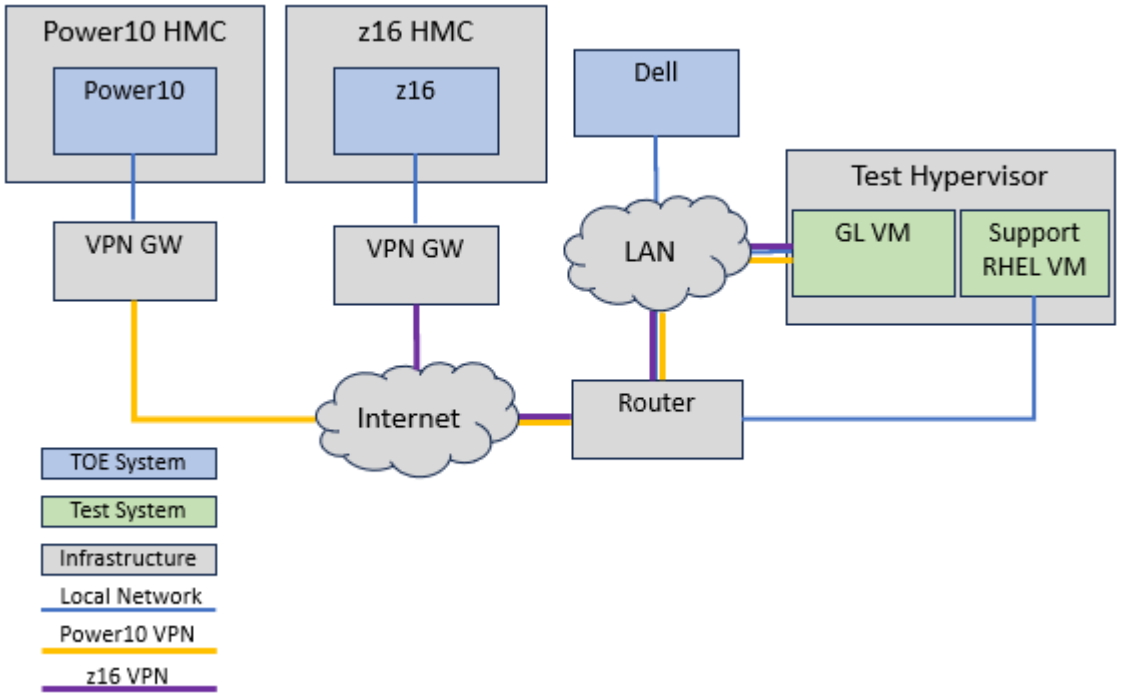

Figure 1 - Test setup

2.4.1 Logging

12 The TOE does not claim to have external logging and only root can access audit logs locally on the TOE.

2.4.2 Time

13 The GL VM was used as NTP server utilizing its system time. The TOEs synchronize time with the GL VM. GL VM and the TOEs are in the same EDT time zone.

2.4.3 Systems

Table 5: Test Systems

Name / HW / SW	Description / Functions	Test Tools
Dell HW: Dell SW: RHEL 9.0 EUS	Fully tested TOE model on Dell platform	bm-search v1.0 annocheck v10.54-2.el9 fmem v1.6.0
z16 HW: Z16 HMC SW: RHEL 9.0 EUS	Fully tested TOE model on z16 platform	bm-search v1.0 annocheck v10.54-2.el9 fmem v1.6.0
Power10 HW: Power10 HMC	Fully tested TOE model on Power10 platform	bm-search v1.0 annocheck v10.54-2.el9

Name / HW / SW	Description / Functions	Test Tools
SW: RHEL 9.0 EUS		fmem v1.6.0
Management Workstation (GL VM) HW: Test Hypervisor SW: Kali GNU/Linux Rolling 2022.1	Protocol Test Host (TLS/SSH) SSH Client Certification Authority Perform Packet Captures NTP Server HTTP/HTTPs Server	Greenlight 3.0.35 OpenSSL 1.1.1m OpenSSH 8.8p1 Debian-1 tcpdump 4.99.1 ntpd 4.2.8 apache2 2.4.52 bm-search 1.0
Support RHEL VM HW: Test Hypervisor SW: RHEL 9.0 EUS	Test repository management	createrepo 0.17.7-4.el9_0
z16 HMC HW: z16 3931-A01 SW: 2.16.0	Host for the z16 TOE	
Power10 HMC HW: POWER10 9080-HEX SW: HCMv10 V10R3	Host for the Power10 TOE	
Test Hypervisor HW: Dell PowerEdge R440 SW: ESXi 7.0.3	Management Workstation and Support RHEL VM Hypervisor	

2.5 Test Platform Equivalency

14 The Security Target claims the following TOE software and compute platforms in the evaluation:

1. Red Hat Enterprise Linux 9.0 EUS

15 on:

1. Intel Xeon Silver x86-64 UEFI platforms (of Cascade Lake microarchitecture)
2. IBM z16 (LPAR) platforms with UEFI 2.3.1 or later.
3. Power10 PowerVM (LPAR) platforms with UEFI 2.3.1 or later

16 The TOE was tested on the following:

1. Dell PowerEdge R440 Xeon Silver 4216 (Cascade Lake)
2. IBM z16 3931-A01 IBM z16

3. IBM POWER10 9080-HEX Power10

2.5.1 Software Differences

17 None. The TOE provides the same security functionality on all platforms.

2.5.2 Hardware Differences

18 The hardware platforms are broken down as follows:

Platform	Architecture	Microarchitecture
Intel Xeon Silver x86-64 UEFI platforms (of Cascade Lake microarchitecture)	x86-64	Cascade Lake
IBM z16 (LPAR) platforms with UEFI 2.3.1 or later.	z/s390x	Generation 16
Power10 PowerVM (LPAR) platforms with UEFI 2.3.1 or later	Power	Generation 10

19 For an OS that is directly interacting with the CPU, the processor architecture is the primary equivalency consideration. The tested systems include each claimed processor architecture. Additionally, due to Policy 5 equivalency considerations, each Microarchitecture was fully tested, providing additional assurance.

2.5.3 Equivalency Conclusion

20 The TOE was fully tested on each processor architecture, with one exception. Due to the VPN performance, it was not possible to perform FCS_SSH_EXT.1.8 data-based rekey testing on the Power10 system. Counting bytes was not deemed to be an architecture specific function, so the Power10 platform is equivalent to the z16 and x86_64 platforms for this specific function.

3 Evaluation Activities for SFRs

3.1 Cryptographic Support (FCS)

3.1.1 NIAP Policy 5

21 To demonstrate that all cryptographic requirements are satisfied, the Assurance Activity Report must clearly indicate all SFRs for which a CAVP certificate is claimed and include, at a minimum, the cryptographic operation, the NIST standard, the SFR supported, the CAVP algorithm list name (e.g. AES, KAS, CVL, etc.) and the CAVP Certificate number.

SFR	Cryptographic Operation	NIST Standard	CAVP Certificate (Algorithm)
FCS_CKM.1	Asymmetric Key Generation: RSA schemes using cryptographic key sizes of 3072-bit or greater that meet the following: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.3	FIPS PUB 186-4	A4771 (RSA)
FCS_CKM.1	Asymmetric Key Generation: ECC schemes using "NIST curves" P-384 and [P-521] that meet the following: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.4	FIPS PUB 186-4	A4771 (ECDSA)
FCS_CKM.1	Asymmetric Key Generation FFC schemes using [safe primes that meet the following: NIST Special Publication 800-56A Revision 3, “Recommendation for Pair-Wise Key Establishment Schemes"	NIST SP 800-56A Revision 3	See section 3.1.3.1.
FCS_CKM.2	Elliptic curve-based key establishment schemes that meets the following: NIST Special Publication 800-56A Revision 3, “Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography”	NIST SP 800-56A Revision 3	A4771 (KAS-ECC-SSC Sp800-56Ar3)
FCS_CKM.2	Finite field-based key establishment schemes that meets NIST Special Publication 800-56A Revision 3, “Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography”	NIST SP 800-56A Revision 3	See section 3.1.3.1.
FCS_COP.1/ ENCRYPT	Encryption/decryption in accordance with a specified cryptographic algorithm:	FIPS PUB 197	A4771 (AES)

SFR	Cryptographic Operation	NIST Standard	CAVP Certificate (Algorithm)
	• AES-CBC • AES-CTR • AES-GCM and cryptographic key sizes 256-bit	NIST SP 800-38A NIST SP 800-38D	
FCS_COP.1/ SIGN	cryptographic signature services (generation and verification): RSA schemes using cryptographic key sizes of [2048-bit (for secure boot only) or greater] that meet the following: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Section 4	FIPS PUB 186-4	A4771 (RSA)
FCS_COP.1/ SIGN	cryptographic signature services (generation and verification): ECDSA schemes using "NIST curves" P-384 and [P-521] that meet the following: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Section 5	FIPS PUB 186-4	A4771 (ECDSA)
FCS_COP.1/ HASH	cryptographic hashing services in accordance with a specified cryptographic algorithm • SHA-256 • SHA-384 • SHA-512	FIPS PUB 180-4	A4771 (SHA2)
FCS_COP.1/ HASH	cryptographic hashing services in accordance with a specified cryptographic algorithm • SHA-512	FIPS PUB 180-4	A4770 (SHA2)
FCS_COP.1/ KEYHMAC	keyed-hash message authentication in accordance with a specified cryptographic algorithm • SHA-256 • SHA-384 • SHA-512	FIPS PUB 198-1	A4771 (HMAC-SHA2)
FCS_COP.1/ KEYHMAC	keyed-hash message authentication in accordance with a specified cryptographic algorithm • SHA-512	FIPS PUB 198-1	A4770 (HMAC-SHA2)
FCS_RBG_EXT.1 / KCAPI	random bit generation services using HMAC_DRBG (any)	NIST SP 800-90A	A4770 (HMAC_DRBG)

SFR	Cryptographic Operation	NIST Standard	CAVP Certificate (Algorithm)
FCS_RBG_EXT.1 / OSSL	random bit generation services using CTR_DRBG (AES)	NIST SP 800-90A	A4771 (CTR_DRBG)

3.1.2 FCS_CKM.1 Cryptographic Key Generation (Refined)

This section has been modified by TD 0712.

3.1.2.1 TSS

22 The evaluator will ensure that the TSS identifies the key sizes supported by the OS. If the ST specifies more than one scheme, the evaluator will examine the TSS to verify that it identifies the usage for each scheme. If "P-256" is selected, the evaluator will examine the TSS to verify that it is only used for Bluetooth functions.

Findings: Section 6.2.1 of the [ST] provides the key sizes supported by the OS. The TOE implements several distinct schemes: FFC using safe primes as specified in NIST SP 800-56A Revision 3, RSA (with sizes 3072 and 4096 bits) and ECC with NIST P-curves P-384 and P521.

The table in section 6.2.1 provides a mapping for each usage defined as "key agreement and key establishment" back to the key column itself which identifies the source of the key and its intended usage. This table is sufficient to be able to determine which keys are used for which services.

The [ST] does not select "P-256."

3.1.2.2 Guidance

23 The evaluator will verify that the AGD guidance instructs the administrator how to configure the OS to use the selected key generation scheme(s) and key size(s) for all uses defined in this PP.

Findings: Section 5.1.1 of the [AGD] describes how to generate persistent asymmetric SSH private keys. Section 6.10 of the [AGD] indicates asymmetric key generation for TLS does not need to be configured.

3.1.2.3 Tests

24 Evaluation Activity Note: The following tests may require the vendor to furnish a developer environment and developer tools that are typically not available to end-users of the OS.

25 The following content should be included if:

- RSA schemes is selected from FCS_CKM.1.1

Key Generation for FIPS PUB 186-4 RSA Schemes

The evaluator will verify the implementation of RSA Key Generation by the OS 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 will 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 will have the TSF generate 10 keys pairs for each supported key length nlen and verify:

- $n = p \cdot q$,
- p and q are probably prime according to Miller-Rabin tests,
- $GCD(p-1, e) = 1$,
- $GCD(q-1, e) = 1$,
- $2^{16} \leq e \leq 2^{256}$ and e is an odd integer,
- $|p-q| > 2^{nlen/2-100}$,
- $p \geq 2^{nlen/2-1/2}$,
- $q \geq 2^{nlen/2-1/2}$,
- $2^{(nlen/2)} < d < LCM(p-1, q-1)$,
- $e \cdot d = 1 \, mod \, LCM(p-1, q-1)$.

26 The following content should be included if:

- ECC schemes using "NIST curves" is selected from FCS_CKM.1.1

Key Generation for Elliptic Curve Cryptography (ECC)

FIPS 186-4 ECC Key Generation Test

For each supported NIST curve, i.e., P-256, P-384 and P-521, the evaluator will 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 will submit the generated key pairs to the public key verification (PKV) function of a known good implementation.

FIPS 186-4 Public Key Verification (PKV) Test

For each supported NIST curve, i.e., P-256, P-384 and P-521, the evaluator will 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 will obtain in response a set of 10 PASS/FAIL values.

27 The following content should be included if:

- FIPS PUB 186-4 is selected from FCS_CKM.1.1

Key Generation for Finite-Field Cryptography (FFC)

The evaluator will 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 $1 \le x \le q-1$
    - len(q) + 64 bit output of RBG, followed by a mod q-1 operation where $1 \le x \le q-1$

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 will have the TSF generate 25 parameter sets and key pairs. The evaluator will 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 !=0,1
- q divides p-1
- $g^q mod\ p = 1$
- $g^x mod\ p = y$

for each FFC parameter set and key pair.

Findings:	CAVP Cert A4771 shows the TOE correctly generates the following asymmetric keys: - RSA 3072 and 4096 - ECDSA P-384 and P-521 The [ST] does not select "FIPS PUB 186-4" for FFC key generation.

3.1.3 FCS_CKM.2 Cryptographic Key Establishment (Refined)

3.1.3.1 Tests

28 The evaluator will ensure that the supported key establishment schemes correspond to the key generation schemes identified in FCS_CKM.1.1 If the ST specifies more than one scheme, the evaluator will examine the TSS to verify that it identifies the usage for each scheme.

Findings:	*This section includes a TSS activity.* Section 6.2.1 of the [ST] provides the supported key establishment schemes claimed in the evaluation. The evaluator considered each of the key establishment schemes that are needed for SSH and TLS and found that they are consistent with the claimed set in the SFR as well as the TSS. Specifically, the [ST] claims finite-field Diffie-Hellman key establishment using safe-prime groups 16 and 18 for SSH and can handle DHE key parameters from TLS servers in the operational environment. The [ST] also claims support for elliptic curve DHE using key sizes P-384 and P-521 which is consistent with both claims in FCS_SSH*_EXT.1 and FCS_TLSC_EXT.1. The table in section 6.2.1 provides a mapping for each usage defined as "key agreement and key establishment" back to the key column itself which identifies the source of the key and its intended usage. This table is sufficient to be able to determine which keys are used for which services.

29 The evaluator will verify that the AGD guidance instructs the administrator how to configure the OS to use the selected key establishment scheme(s).

Findings:	*This section includes a Guidance activity.* Section 5.1 of the [AGD] describes how to configure the SSH key agreement schemes. Section 6.10 of the [AGD] indicates key agreement for TLS does not need to be configured.

30 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

31 The evaluator will verify the implementation of the key establishment schemes supported by the OS using the applicable tests below.

32 The following content should be included if:

- Elliptic curve-based, Finite field-based is selected from FCS_CKM.2.1

SP800-56A Key Establishment Schemes

The evaluator will verify the OS'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 the OS has implemented the components of the key agreement scheme according to the specifications in the Recommendation. These components include the calculation of the discrete logarithm cryptography (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 will 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 MAC data and the calculation of MAC tag.

Function Test

The Function test verifies the ability of the OS to implement the key agreement schemes correctly. To conduct this test the evaluator will generate or obtain test vectors from a known good implementation of the OS's 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 will generate 10 sets of test vectors. The data set consists of one set of domain parameter values (FCC) 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 will obtain the DKM, the corresponding OS's public keys (static and/or ephemeral), the MAC tag(s), and any inputs used in the KDF, such as the Other Information field OI and OS id fields.

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

The evaluator will 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 OS will perform the above for each implemented approved MAC algorithm.

Validity Test

The Validity test verifies the ability of the OS to recognize another party's valid and invalid key agreement results with or without key confirmation. To conduct this test, the evaluator will obtain a list of the supporting cryptographic functions included in the SP800-56A Revision 3 key agreement

implementation to determine which errors the OS should be able to recognize. The evaluator generates a set of 24 FCC or 30 ECC test vectors consisting of data sets including domain parameter values or NIST approved curves, the evaluator's public keys, the OS's public/private key pairs, MAC tag, and any inputs used in the KDF, such as the other info and OS id fields.

The evaluator will inject an error in some of the test vectors to test that the OS recognizes invalid key agreement results caused by the following fields being incorrect: the shared secret value Z, the DKM, the other information field OI, the data to be MACed, or the generated MAC tag. If the OS 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 OS's static private key to assure the OS 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 will remain unmodified and therefore should result in valid key agreement results (they should pass).

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

Findings:	CAVP Cert A4771 shows the TOE correctly implements ECDH key agreement using P-384 and P-521.

33 The following content should be included if:

- RSA-based is selected from FCS_CKM.2.1

RSAES-PKCS1-v1_5 Key Establishment Schemes

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

Findings:	The TOE does not claim the use of RSAES-PKCS1-v1_5 key establishment.

34 The following content should be included if:

- Finite field-based is selected from FCS_CKM.2.1

FFC Schemes using "safe-prime" groups (identified in Appendix D of SP 800-56A Revision 3)

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

High-Level Test Description
Verify each safe-prime group is successfully negotiated with a known good implementation of each protocol.

High-Level Test Description
Findings: PASS – The evaluator confirmed the TOE successfully negotiated each safe-prime group with a known good implementation of each protocol.

3.1.4 FCS_CKM_EXT.4 Cryptographic Key Destruction

This section has been modified by TD 0701.

3.1.4.1 TSS

35 The evaluator examines the TSS 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.

Findings: Section 6.2.3 of the [ST] provides a description of how key material is overwritten when stored in volatile memory. The mechanisms are consistent with the SFR in section 5.3.2 of the [ST].

Table 16 in the [ST] includes a description of where the specific keys reside and the mechanism by which they are destroyed (referring back to the zeroization process description given in section 6.2.3 of the [ST]).

The description appears to be complete and accurate.

36 The evaluator will check to ensure the TSS 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).

Findings: Section 6.2.3 of the [ST] provides a description of how key material is overwritten when stored in non-volatile memory. The mechanisms are consistent with the SFR in section 5.3.2 of the [ST].

Table 16 in the [ST] includes a description of where the specific keys reside and the mechanism by which they are destroyed (referring back to the zeroization process description given in section 6.2.3 of the [ST]).

The description appears to be complete and accurate.

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

Findings: The [ST] in section 5.3.2 selects the open assignment for the non-volatile overwrite pattern. The pattern is obtained from /dev/urandom which produces a pseudo random bit stream that does not contain CSPs..

38 The evaluator will check that the TSS identifies any configurations or circumstances that may not strictly conform to the key destruction requirement.

Findings:	The [ST] in section 5.3.2 does not identify any configuration or circumstances that may not strictly conform to the key destruction requirement.

39 If the selection *"destruction of all key encrypting keys (KEKs) protecting the target key according to FCS_CKM_EXT.4.1, where none of the KEKs protecting the target key are derived"* is included the evaluator will examine the TOE's keychain in the TSS and identify each instance when a key is destroyed by this method. In each instance the evaluator will 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 will 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.

Findings:	The [ST] in section 5.3.2 does not make this selection.

3.1.4.2 Guidance

40 There are a variety of concerns that may prevent or delay key destruction in some cases. The evaluator will 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 will 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.

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

42 When the TOE does not have full access to the physical memory, it is possible that the storage may be implementing wear-leveling 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.

43 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.

44 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.

Findings:	[AGD] Section 6.12.1 provide a warning that all copies of a key might not be destroyed if the sector is replaced with a spare sector. The guidance recommends end-of-lifeing drives before a significant amount of damage has occurred to mitigate this risk. [AGD] Section 6.12 notes that the shred function does not work on SSDs. No other instances that may not strictly conform to the key destruction requirement.

3.1.4.3 Tests

- Test 1: Applied to each key held as 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 destruction method key was removal of power, then this test is unnecessary. The evaluator will:

    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. 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.

High-Level Test Description
Perform a TLS connection to a server. Once the connection is complete so the keys are no longer needed, dump the available RAM. Verify the RAM sump does not contain TLS Master Key. It should not be found.
Findings: PASS – The evaluator confirmed that the TLS client library properly sanitizes critical key material in volatile memory.

- Test 2: Applied to each key help in non-volatile memory and subject to destruction by the TOE. The evaluator will 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 data 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.

The test succeeds if step 3 fails.

High-Level Test Description
Destroy each key type held in non-volatile memory. Attempt to use each destroyed key and verify the operation that requires the key fails.
Findings: PASS – The evaluator confirmed cryptographic operations that utilize a specific key fail after the key has been destroyed.

- Test 3:

  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.

  The following tests are used to determine if the TOE is able to request the platform to overwrite the key with a TOE supplied pattern.

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

High-Level Test Description
While performing Test 2, the evaluator recorded the value of each key, caused the TOE to perform normal cryptographic processing using each key, caused the TOE to clear each key, and searched the logical view of the drive for instances of each key.
Findings: PASS – The evaluator confirmed normal cryptographic processing succeeded and no instances of the keys were found after clearing.

- Test 4: Applied to each key held as non-volatile memory and subject to destruction by overwrite by the TOE. The evaluator will 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.

High-Level Test Description
While performing Test 2, the evaluator recorded the storage location of each key, caused the TOE to perform normal cryptographic processing using each key, caused the TOE to clear each key, and read the logical storage location of each key.
Findings: PASS – The evaluator confirmed normal cryptographic processing succeeded and the logical storage location contained pseudo random data after clearing.

3.1.5 FCS_COP.1/ENCRYPT Cryptographic Operation – Encryption/Decryption (Refined)

This section has been modified by TD 0712.

3.1.5.1 TSS

45 If "128-bit" is selected, the evaluator will examine the TSS to verify that 128-bit is only used with AES-CCM for Bluetooth functions.

Findings: Section 5.3.2 of the [ST] does not select 128-bit.

3.1.5.2 Guidance

46 The evaluator will verify that the AGD documents contains instructions required to configure the OS to use the required modes and key sizes.

Findings: SSH uses AES-256-CTR and AES-256-GCM.

TLS uses AES-256-GCM.

Encryption of sensitive data uses AES-256-CTR and AES-256-CBC.

Please see Section 7.1.3.10 in this Assurance Activity Report for SSH server and client findings regarding the configuration of SSH.

Please see Section 7.1.2.2 in this Assurance Activity Report for TLS client findings regarding the configuration of TLS.

[AGD] Section 6.11 describes how to invoke the encryption of sensitive data using the required modes and key size.

3.1.5.3 Tests

47 The evaluator will execute all instructions as specified to configure the OS to the appropriate state. The evaluator will perform all of the following tests for each algorithm implemented by the OS and used to satisfy the requirements of this PP:

48 The following content should be included if:

- AES-XTS is selected from FCS_COP.1.1/ENCRYPT

XTS-AES Test

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

- 512 bit (for AES-256) key
- Three data unit (i.e., plaintext) lengths. One of the data unit lengths will be a nonzero integer multiple of 256 bits, if supported. One of the data unit lengths will be an integer multiple of 256 bits, if supported. The third data unit length will be either the longest supported data unit length or 216 bits, whichever is smaller.

Using a set of 100 (key, plaintext and 256-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 will 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 XTSAES decrypt.

Findings:	Section 5.3.2 of the [ST] does not select AES-XTS.

49 The following content should be included if:

- AES-CBC is selected from FCS_COP.1.1/ENCRYPT

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 256-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.

- Test 5: To test the encrypt functionality of AES-CBC, the evaluator will supply a set of 5 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. The plaintext values will encrypted with a 256-bit all-zeros key. To test the decrypt functionality of AES-CBC, the evaluator will perform the same test as for encrypt, using 5 ciphertext values as input and AES-CBC decryption.

- Test 6: To test the encrypt functionality of AES-CBC, the evaluator will supply a set of five 256-keys 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. To test the decrypt functionality of AES-CBC, the evaluator will perform the same test as for encrypt, using an all-zero ciphertext value as input and AES-CBC decryption.

- Test 7: To test the encrypt functionality of AES-CBC, the evaluator will supply the a 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. Key i will 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 will supply the set 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 set of key/ciphertext pairs will have 256 256-bit key/ciphertext pairs. Key i in each set will 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 will be the value that results in an all-zeros plaintext when decrypted with its corresponding key.

- Test 8: To test the encrypt functionality of AES-CBC, the evaluator will supply the set of 256 plaintext values described below and obtain the ciphertext values that result from AES-CBC encryption of the given plaintext using a 256-bit key value of all zeros with an IV of all zeros. Plaintext value i in each set will have the leftmost i bits be ones and the rightmost 256-i bits be zeros, for i in [1,256].

To test the decrypt functionality of AES-CBC, the evaluator will 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.

Findings:	CAVP Cert A4771 shows the TOE correctly implements AES-CBC-256.

50 The following content should be included if:

- AES-CTR is selected from FCS_COP.1.1/ENCRYPT

AES-CTR Test

Known Answer Tests (KATs)

There are four Known Answer Tests (KATs) described below. For all KATs, the plaintext, initialization vector (IV), and ciphertext values shall be 256-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 will compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

- Test 9: To test the encrypt functionality, the evaluator will supply 5 plaintext values and obtain the ciphertext value that results from encryption of the given plaintext using a 256-bit key value of all zeros and an IV of all zeros. To test the decrypt functionality, the evaluator will perform the same test as for encrypt, using the 5 ciphertext values as input.

- Test 10: To test the encrypt functionality, the evaluator will supply 5 256-bit 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. To test the decrypt functionality, the evaluator will perform the same test as for encrypt, using an all zero ciphertext value as input.

- Test 11: To test the encrypt functionality, the evaluator will supply a set 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 and an IV of all zeros. The set of keys shall have shall have 256 256-bit keys. Keyi shall have the leftmost i bits be ones and the rightmost 256-i bits be zeros, for i in [1, N]. To test the decrypt functionality, the evaluator will supply the set 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 set of key/ciphertext pairs shall have 256 256-bit pairs. Keyi shall have the leftmost i bits be ones and the rightmost 256-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.

- Test 12: To test the encrypt functionality, the evaluator will supply the set of 256 plaintext values described below and obtain the two ciphertext values that result from encryption of the given plaintext 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 256-i bits be zeros, for i in [1, 256]. To test the decrypt functionality, the evaluator will perform the same test as for encrypt, using ciphertext values of the same form as the plaintext in the encrypt test as input.

Multi-Block Message Test

The evaluator will 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 will 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 will 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 will 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.

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 will test the encrypt functionality using 100 plaintext/key pairs. Each key shall be 256-bit. The plaintext values shall be 256-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.

Findings:	CAVP Cert A4771 shows the TOE correctly implements AES-CTR-256.

51 The following content should be included if:

- AES Key Wrap (KW) (as defined in NIST SP 800-38F), AES Key Wrap with Padding (KWP) (as defined in NIST SP 800-38F) is selected from FCS_COP.1.1/ENCRYPT

AES Key Wrap (AES-KW) and Key Wrap with Padding (AES-KWP) Test

The evaluator will test the authenticated encryption functionality of AES-KW for EACH combination of the following input parameter lengths:

- 256 bit key encryption keys (KEKs)
- Three plaintext lengths. One of the plaintext lengths will be two semi-blocks (256 bits). One of the plaintext lengths will be three semi-blocks (192 bits). The third data unit length will be the longest supported plaintext length less than or equal to 64 semi-blocks (4096 bits).

using a set of 100 key and plaintext pairs and obtain the ciphertext that results from AES-KW authenticated encryption. To determine correctness, the evaluator will use the AES-KW authenticated-encryption function of a known good implementation.

The evaluator will test the authenticated-decryption functionality of AES-KW using the same test as for authenticated-encryption, replacing plaintext values with ciphertext values and AES-KW authenticated-encryption with AES-KW authenticated-decryption.

Findings:	Section 5.3.2 of the [ST] does not select AES-KW.

52 The following content should be included if:

- AES Key Wrap with Padding (KWP) (as defined in NIST SP 800-38F) is selected from FCS_COP.1.1/ENCRYPT

The evaluator will test the authenticated-encryption functionality of AES-KWP using the same test as for AES-KW authenticated-encryption with the following change in the three plaintext lengths:

- One plaintext length will be one octet. One plaintext length will be 20 octets (160 bits).
- One plaintext length will be the longest supported plaintext length less than or equal to 512 octets (4096 bits).

The evaluator will test the authenticated-decryption functionality of AES-KWP using the same test as for AES-KWP authenticated-encryption, replacing plaintext values with ciphertext values and AES-KWP authenticated-encryption with AES-KWP authenticated-decryption.

Findings:	Section 5.3.2 of the [ST] does not select AES-KWP.

53 The following content should be included if:

- AES-CCM or AES-CCMP-256 is selected from FCS_COP.1.1/ENCRYPT

AES-CCM Tests

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

- 128 bit (if selected) and 256 bit keys
- Two payload lengths. One payload length will be the shortest supported payload length, greater than or equal to zero bytes. The other payload length will be the longest supported payload length, less than or equal to 32 bytes (256 bits).
- Two or three associated data lengths. One associated data length will be 0, if supported. One associated data length will be the shortest supported payload length, greater than or equal to zero bytes. One associated data length will be the longest supported payload length, less than or equal to 32 bytes (256 bits). If the implementation supports an associated data length of 2 16 bytes, an associated data length of 216 bytes will be tested.
- Nonce lengths. The evaluator will test all nonce lengths between 7 and 13 bytes, inclusive, that are supported by the OS.
- Tag lengths. The evaluator will test all of the following tag length values that are supported by the OS: 4, 6, 8, 10, 12, 14 and 16 bytes.

To test the generation-encryption functionality of AES-CCM, the evaluator will perform the following four tests:

- Test 13: For EACH supported key and associated data length and ANY supported payload, nonce and tag length, the evaluator will supply one key value, one nonce value and 10 pairs of associated data and payload values and obtain the resulting ciphertext.

- Test 14: For EACH supported key and payload length and ANY supported associated data, nonce and tag length, the evaluator will supply one key value, one nonce value and 10 pairs of associated data and payload values and obtain the resulting ciphertext.

- Test 15: For EACH supported key and nonce length and ANY supported associated data, payload and tag length, the evaluator will supply one key value and 10 associated data, payload and nonce value 3-tuples and obtain the resulting ciphertext.

- Test 16: For EACH supported key and tag length and ANY supported associated data, payload and nonce length, the evaluator will supply one key value, one nonce value and 10 pairs of associated data and payload values and obtain the resulting ciphertext.

To determine correctness in each of the above tests, the evaluator will compare the ciphertext with the result of generation-encryption of the same inputs with a known good implementation.

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 will supply a key value and 15 nonce, associated data and ciphertext 3-tuples and obtain either a FAIL result or a PASS result with the decrypted payload. The evaluator will supply 10 tuples that should FAIL and 5 that should PASS per set of 15.

Additionally, the evaluator will use tests from the IEEE 802.11-02/362r6 document "Proposed Test vectors for IEEE 802.11 TGi", dated September 10, 2002, Section 2.1 AESCCMP Encapsulation Example and Section 2.2 Additional AES CCMP Test Vectors to further verify the IEEE 802.11-2007 implementation of AES-CCMP.

Findings:	The ST does not claim use of AES-CCM or AES-CCMP.

54 The following content should be included if:

- AES-GCMP-256 is selected from FCS_COP.1.1/ENCRYPT

AES-GCMP Test

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

- 256 bit keys

- Two plaintext lengths. One of the plaintext lengths will be a non-zero integer multiple of 256 bits, if supported. The other plaintext length will not be an integer multiple of 256 bits, if supported.

- Three AAD lengths. One AAD length will be 0, if supported. One AAD length will be a non-zero integer multiple of 256 bits, if supported. One AAD length will not be an integer multiple of 256 bits, if supported.

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

The evaluator will 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 will 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 will 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 will 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 will compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

AES-GCMP Monte Carlo Tests

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

- 256 bit keys
- Two plaintext lengths. One of the plaintext lengths will be a non-zero integer multiple of 256 bits, if supported. The other plaintext length will not be an integer multiple of 256 bits, if supported.
- Three AAD lengths. One AAD length will be 0, if supported. One AAD length will be a non-zero integer multiple of 256 bits, if supported. One AAD length will not be an integer multiple of 256 bits, if supported.
- Two IV lengths. If 96 bit IV is supported, 96 bits will be one of the two IV lengths tested.

The evaluator will 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 will 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 will 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 will 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 will compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

Findings:	Section 5.3.2 of the [ST] does not select AES-GCMP.

Note:	[PP_OS_v4.3] does not specify any tests for AES-GCM; however, CAVP Cert. A4771 shows the TOE correctly implements AES-GCM.

3.1.6 FCS_COP.1/HASH Cryptographic Operation – Hashing (Refined)

3.1.6.1 Tests

56 The evaluator will 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.

Findings:	*This section includes a TSS activity.* The evaluator found that the hashing algorithms are associated with other cryptographic functions. In the [ST] in section 6.2.6, hashing algorithms are associated with digital signatures. The hashing algorithms in section 6.2.5 are consistent with those used for FCS_COP.1/SIGN. In section 6.2.7 of the [ST], the hashing algorithms are mapped to where they are used in HMACs. The hashing algorithms in section 6.2.5 are consistent with those used for FCS_COP.1/KEYHMAC. In section 6.2.8 of the [ST], the TOE makes use of an HMAC_DRBG which requires a hashing function. The hashing function is SHA2-512 and is consistent with the claims made in section 6.2.5. The SSH protocol makes use of hashing functions and digital signatures. The evaluator considered each of the algorithms described in sections 6.2.11 of the [ST] and found they were consistent with the hashing algorithms claimed in 6.2.5. The TLS protocol makes use of hashing functions and digital signatures. The evaluator considered each of the algorithms described in sections 6.2.12 of the [ST] and found they were consistent with the hashing algorithms claimed in 6.2.5. Boot integrity makes use of digital signatures and hashing functions. The hashing function in section 6.6.5 of the [ST] is consistent with those in section 6.2.5. Trusted updates for both OS and application software make use of digital signatures and hashing. The hashing algorithms in section 6.6.6 of the [ST] is consistent with the hashing algorithms in section 6.2.5.

57 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 only hashes 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 test MACs. The evaluator will perform all of the following tests for each hash algorithm implemented by the TSF and used to satisfy the requirements of this PP.

58 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 17: Short Messages Test (Bit oriented Mode) - The evaluator will generate 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 will be pseudorandomly generated. The evaluator will 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 18: Short Messages Test (Byte oriented Mode) - The evaluator will generate 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 will be pseudorandomly generated. The evaluator will 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 19: Selected Long Messages Test (Bit oriented Mode) - The evaluator will generate 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 will be pseudorandomly generated. The evaluator will 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 20: Selected Long Messages Test (Byte oriented Mode) - The evaluator will generate 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 will be pseudorandomly generated. The evaluator will 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 21: Pseudorandomly Generated Messages Test - This test is for byte-oriented implementations only. The evaluator will 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 evaluator will then formulate a set of 100 messages and associated digests by following the algorithm provided in Figure 1 of [SHAVS]. The evaluator will then ensure that the correct result is produced when the messages are provided to the TSF.

Findings:	CAVP Cert. A4771 shows the TOE correctly implements SHA-256, SHA-384, and SHA-512. CAVP Cert. A4770 shows the TOE correctly implements SHA-512.

3.1.7 FCS_COP.1/SIGN Cryptographic Operation – Signing (Refined)

This section has been modified by TD 0809.

3.1.7.1 TSS

59 [Conditional: if “2048-bit (for secure boot only) or greater” is selected] The evaluator shall check that the TSS documents that 2048-bit RSA is used only for secure boot and a greater key size is used for any other functions.

Findings:	Sections 6.2.6 and 6.6.5 of the [ST] clearly state that RSA 2048 is only used for secure boot. All other RSA signatures specified in the TSS used RSA 3072 or 4096.

3.1.7.2 Guidance

60 [Conditional: if “2048-bit (for secure boot only) or greater” is selected] The evaluator shall check that the AGD documents any configuration needed to ensure 2048-bit RSA is used only for secure boot and a greater key size is used for any other functions.

Findings:	Section 5.1 of the [AGD] describes how to configure the SSH signature algorithms. Section 6.10 of the [AGD] indicates signature algorithms for TLS does not need to be configured. [AGD] indicates SSH and TLS do not use RSA 2048. Section 3.2 of the [AGD] indicates updates use RSA 4096. Secure boot signatures are generated by the vendor using the key chain specified in [ST] Section 6.6.5. The user does not have control over these keys and key sizes. Since SSH, TLS, and updates do no use RSA 2048-bit keys, no configuration is needed to ensure 2048-bit RSA is only used for secure boot.

3.1.7.3 Tests

61 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.

62 The following content should be included if:

- ECDSA schemes is selected from FCS_COP.1.1/SIGN

ECDSA Algorithm Tests

- Test 22: ECDSA FIPS 186-4 Signature Generation Test. For each supported NIST curve (i.e., P-384 and P-521) and SHA function pair, the evaluator will 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 will use the signature verification function of a known good implementation.

- Test 23: ECDSA FIPS 186-4 Signature Verification Test. For each supported NIST curve (i.e., P-384 and P-521) and SHA function pair, the evaluator will 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 will verify that 5 responses indicate success and 5 responses indicate failure.

Findings:	CAVP Cert. A4771 shows the TOE correctly implements ECDSA P-384 and P-521.

63 The following content should be included if:

- RSA schemes using cryptographic key sizes of 2048-bit or greater that meet the following: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Section 4 is selected from FCS_COP.1.1/SIGN

RSA Signature Algorithm Tests

- Test 24: Signature Generation Test. The evaluator will verify the implementation of RSA Signature Generation by the OS 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 will have the OS use its private key and modulus value to sign these messages. The evaluator will verify the correctness of the TSF' signature using a known good implementation and the associated public keys to verify the signatures.

- Test 25: Signature Verification Test. The evaluator will perform the Signature Verification test to verify the ability of the OS to recognize another party's valid

and invalid signatures. The evaluator will 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 evaluator will verify that the OS returns failure when validating each signature.

Findings:	CAVP Cert. A4771 shows the TOE correctly implements RSA 2048, 3072, and 4096.

3.1.8 FCS_COP.1/KEYHMAC Cryptographic Operation - Keyed-Hash Message Authentication (Refined)

3.1.8.1 Tests

64 The evaluator will perform the following activities based on the selections in the ST.

65 For each of the supported parameter sets, the evaluator will compose 15 sets of test data. Each set consists of a key and message data. The evaluator will have the OS generate HMAC tags for these sets of test data. The resulting MAC tags will be compared against the result of generating HMAC tags with the same key using a known-good implementation.

Findings:	CAVP Cert. A4771 shows the TOE correctly implements HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512. CAVP Cert. A4770 shows the TOE correctly implements HMAC-SHA-512.

3.1.9 FCS_RBG_EXT.1/KCAPI, FCS_RBG_EXT.1/OSSL Random Bit Generation

3.1.9.1 Tests

66 The evaluator will perform the following tests:

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

Findings:	*This section includes a Guidance activity.* [AGD] section 4.2 instructs the user to enable FIPS mode to ensure the system generates all keys (i.e., RNG functionality) using FIPS approved algorithms. This is the only configuration that is necessary to configure 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 will 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 will 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 list contains 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 <= 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.

Findings:	CAVP Cert. A4770 shows the TOE correctly implements HMAC_DRBG(SHA-512). CAVP Cert. A4771 shows the TOE correctly implements CTR_DRBG(AES-256).

67 Documentation will be produced - and the evaluator will perform the activities - in accordance with Appendix E - Entropy Documentation and Assessment and the Clarification to the Entropy Documentation and Assessment Annex.

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

Findings:	An Entropy Assessment Report was submitted and evaluated in accordance with Appendix E of the PP as part of this evaluation.

3.1.10 FCS_STO_EXT.1 Storage of Sensitive Data

3.1.10.1 TSS

69 The evaluator will check the TSS to ensure that it lists all persistent sensitive data for which the OS provides a storage capability. For each of these items, the evaluator will confirm that the TSS lists for what purpose it can be used, and how it is stored. The evaluator will confirm that cryptographic operations used to protect the data occur as specified in FCS_COP.1/ENCRYPT.

Findings:	Section 6.2.10 of the [ST] provides the definition for “sensitive data” which the OS provides a secure storage capability. Sensitive data is protected through filesystem permissions and/or encryption using AES-256 CBC or CTR.

3.1.10.2 Guidance

70 The evaluator will consult the developer documentation to verify that instructions exists on applications should securely store credentials.

Findings:	[AGD] Section 6.11 instructs application developers to store credentials in /etc and provides the ability to encrypt sensitive data.

3.2 User Data Protection (FDP)

3.2.1 FDP_ACF_EXT.1 Access Controls for Protecting User Data

3.2.1.1 TSS

71 The evaluator will confirm that the TSS comprehensively describes the access control policy enforced by the OS. The description must include the rules by which accesses to particular files and directories are determined for particular users. The evaluator will inspect the TSS to ensure that it describes the access control rules in such detail that given any possible scenario between a user and a file governed by the OS the access control decision is unambiguous.

Findings:	Section 6.3.1 of the [ST] contains a summary of the standard Unix permission bits as well as ACLs used in the TOE to achieve discretionary access of files. The material needed to satisfy the requirement that the TSS provide a completely unambiguous decision for any scenario between a user and a file is beyond the scope of the TSS. Instead, the ST author includes a reference to [RHEL] which is the authoritative means by which such decisions are governed.

3.2.1.2 Tests

72 The evaluator will create two new standard user accounts on the system and conduct the following tests:

- Test 26: The evaluator will authenticate to the system as the first user and create a file within that user's home directory. The evaluator will then log off the system and log in as the second user. The evaluator will then attempt to read the file created in the first user's home directory. The evaluator will ensure that the read attempt is denied.

High-Level Test Description
Create a file as user1 in the user's home directory. Then log into the TOE as user2 and attempt to read the contents of the file just created in user1's home directory. The attempt should fail. As user2, use a kernel system call to attempt to read the file directly. The attempt should fail.
Findings: PASS – The evaluator confirmed that user2 is unable to read the file created by user1 due to file access control permissions.

- Test 27: The evaluator will authenticate to the system as the first user and create a file within that user's home directory. The evaluator will then log off the system and log in as the second user. The evaluator will then attempt to

modify the file created in the first user's home directory. The evaluator will ensure that the modification is denied.

High-Level Test Description
Building upon the previous test case, log into the TOE as user2 and attempt to modify the file created by user1 in the first test case. Show this attempt fails.
Findings: PASS – The evaluator confirmed that user2 is unable to write to the file created by user1 due to file access control permissions.

- Test 28: The evaluator will authenticate to the system as the first user and create a file within that user's user directory. The evaluator will then log off the system and log in as the second user. The evaluator will then attempt to delete the file created in the first user's home directory. The evaluator will ensure that the deletion is denied.

High-Level Test Description
Building upon the previous test case, log into the TOE as user2 and attempt to delete the file created by user1 in the first test case. Show this attempt fails.
Findings: PASS – The evaluator confirmed that user2 is unable to delete the file created by user1 due to file access control permissions.

- Test 29: The evaluator will authenticate to the system as the first user. The evaluator will attempt to create a file in the second user's home directory. The evaluator will ensure that the creation of the file is denied.

High-Level Test Description
Building upon the previous test case, log into the TOE as user2 and attempt to create a new file into the home directory of user1. Show this attempt fails.
Findings: PASS – The evaluator confirmed that user2 is unable to create a new file in user1’s directory due to file access control permissions.

- Test 30: The evaluator will authenticate to the system as the first user and attempt to modify the file created in the first user's home directory. The evaluator will ensure that the modification of the file is accepted.

High-Level Test Description
Building upon the previous test case, log into the TOE as user1 and attempt to modify one of their own files. Show this attempt succeeds.
Findings: PASS – The evaluator confirmed that user1 is able to create and modify files in their own directory.

- Test 31: The evaluator will authenticate to the system as the first user and attempt to delete the file created in the first user's directory. The evaluator will ensure that the deletion of the file is accepted.

High-Level Test Description
Building upon the previous test case, log into the TOE as user1 and attempt to delete one of their own files. Show this attempt succeeds.
Findings: PASS – The evaluator confirmed that user1 is able to delete files in their own directory.

3.3 Security management (FMT)

3.3.1 FMT_MOF_EXT.1 Management of security functions behavior

3.3.1.1 TSS

73 The evaluator will verify that the TSS describes those management functions that are restricted to Administrators, including how the user is prevented from performing those functions, or not able to use any interfaces that allow access to that function.

Findings:	Section 6.5.1 claims that any user who is not a member of the "wheel" group cannot perform any of the functions restricted to Administrators as it will be denied via the "sudo" command. The "sudo" command is the primary interface that allows users to elevate their privileges allowing management of the TOE. Based on the table in FMT_SMF_EXT.1 in section 5.3.5 of the [ST], only Administrators are permitted to perform the given functions.

3.3.1.2 Tests

74 The evaluator will also perform the following test.

- Test 32: For each function that is indicated as restricted to the administrator, the evaluation will perform the function as an administrator, as specified in the Operational Guidance, and determine that it has the expected effect as outlined by the Operational Guidance and the SFR. The evaluator will then perform the function (or otherwise attempt to access the function) as a non-administrator and observe that they are unable to invoke that functionality.

High-Level Test Description
For each of the claimed management functions, attempt to modify the setting as an unprivileged user and show that the access attempt meets the claim in the table for FMT_SMF_EXT.1 in the [ST]. An audit log entry should be generated what matches this expected behaviour. For each of the claimed management functions, attempt to modify the setting as a privileged user and show that the access attempt meets the claim in the table for FMT_SMF_EXT.1 in the [ST]. An audit log entry should be generated what matches this expected behaviour. For each successful modification of a configuration option, show that the modification results in behaviour that is expected from the change. For example, when configuring the session timeout, show that the session times out.
Findings: PASS – The evaluator confirmed that the management tasks, as described in the AGD, can be restricted to administrators and result in the expected action when executed.

3.3.2 FMT_SMF_EXT.1 Specification of Management Functions

3.3.2.1 Guidance

75 The evaluator will verify that every management function captured in the ST is described in the operational guidance and that the description contains the information required to perform the management duties associated with the management function.

Findings:	The evaluator mapped each management function claimed in [ST] section 5.3.5 FMT_SMF.1.1 to the section in guidance where the management function is described: • Enable/disable [session timeout] – [AGD] Section 6.5 • Configure [session] inactivity timeout – [AGD] Section 6.5 • Import keys/secrets into the secure key storage – [AGD] Section 6.11 • Configure local audit storage capacity – [AGD] Section 6.7 • Configure minimum password length – [AGD] Section 6.6, “man pwquality conf” on the TOE itself specifies minlen. • Configure minimum number of special characters in password – [AGD] Section 6.6, “man pwquality conf” on the TOE itself specifies ocredit. • Configure minimum number of numeric characters in password – [AGD] Section 6.6, “man pwquality conf” on the TOE itself specifies dcreidt set to a negative value. • Configure minimum number of uppercase characters in password – [AGD] Section 6.6, “man pwquality conf” on the TOE itself specifies ucredit set to a negative value. • Configure minimum number of lowercase characters in password – [AGD] Section 6.6, “man pwquality conf” on the TOE itself specifies lcredit set to a negative value. • Configure lockout policy for unsuccessful authentication attempts through [timeouts between attempts] – [AGD] Section 6.4 • Configure host-based firewall [AGD] Section 6.2 • Configure audit rules – [AGD] Section 5.3.1 • Configure name/address of network time server – [AGD] Section 5.2 • Enable/disable automatic software update – [AGD] Section 3.2.2.3 The evaluator verified that each description includes the information necessary to perform the management duties.

3.3.2.2 Tests

76 The evaluator will test the OS's ability to provide the management functions by configuring the operating system 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.

High-Level Test Description
Testing management functions is performed in FMT_MOF_EXT.1.
Findings: PASS – The evaluator confirmed that the management tasks, as described in the AGD, can be restricted to administrators and result in the expected action when executed.

3.4 Protection of the TSF (FPT)

3.4.1 FPT_ACF_EXT.1 Access controls

3.4.1.1 TSS

77 The evaluator will confirm that the TSS specifies the locations of kernel drivers/modules, security audit logs, shared libraries, system executables, and system configuration files. Every file does not need to be individually identified, but the system's conventions for storing and protecting such files must be specified.

Findings:	Section 6.3.1 of the [ST] provides the locations for: 1) Kernel drivers/modules: /boot, /usr/lib/modules and /usr/lib/firmware 2) Security audit logs: /var/log/audit and /var/log/secure 3) Shared libraries: /usr/lib64 and /usr/lib 4) System executables: /usr/sbin, /usr/bin and /usr/libexec 5) System configuration files: /etc and /usr/lib Section 6.3.1 of the [ST] claims that all of these files are protected using the access control permissions described in the same section.

3.4.1.2 Tests

78 The evaluator will create an unprivileged user account. Using this account, the evaluator will ensure that the following tests result in a negative outcome (i.e., the action results in the OS denying the evaluator permission to complete the action):

- Test 33: The evaluator will attempt to modify all kernel drivers and modules.

High-Level Test Description
Using an unprivileged user account, attempt to modify all kernel drivers and modules and show the attempts are unsuccessful.
Findings: PASS – The evaluator confirmed that unprivileged users are unable to modify kernel drivers and modules in the paths defined in the ST.

- Test 34: The evaluator will attempt to modify all security audit logs generated by the logging subsystem.

High-Level Test Description
For each of the files in the defined log and audit set, attempt to modify them as an unprivileged user.
Findings: PASS – The evaluator confirmed unprivileged users are unable to modify files located in the log directories in the paths defined in the ST.

- Test 35: The evaluator will attempt to modify all shared libraries that are used throughout the system.

High-Level Test Description
For each of the files in the defined shared library locations, attempt to modify them as an unprivileged user.
Findings: PASS – The evaluator confirmed unprivileged users are unable to modify files located in the shared library directories in the paths defined in the ST.

- Test 36: The evaluator will attempt to modify all system executables.

High-Level Test Description
For each of the files in the defined system executable locations, attempt to modify them as an unprivileged user.
Findings: PASS – The evaluator confirmed unprivileged users are unable to modify files located in the system executable directories in the paths defined in the ST.

- Test 37: The evaluator will attempt to modify all system configuration files.

High-Level Test Description
For each of the files in the defined system configuration directories, attempt to modify them as an unprivileged user.
Findings: PASS – The evaluator confirmed unprivileged users are unable to modify files located in the system configuration directories in the paths defined in the ST.

- Test 38: The evaluator will attempt to modify any additional components selected.

Test Not Applicable:	The [ST] does not select any other components to be checked.

79 The evaluator will create an unprivileged user account. Using this account, the evaluator will ensure that the following tests result in a negative outcome (i.e., the action results in the OS denying the evaluator permission to complete the action):

- Test 39: The evaluator will attempt to read security audit logs generated by the auditing subsystem

High-Level Test Description
For each of the files defined as auditing and logging files, attempt to read them as an unprivileged user.
Findings: PASS – The evaluator confirmed unprivileged users are unable to read files located in the log directories in the paths defined in the ST.

- Test 40: The evaluator will attempt to read system-wide credential repositories

High-Level Test Description
For each of the files defined as system-wide credential stores, attempt to read them as an unprivileged user.
Findings: PASS – The evaluator confirmed unprivileged users are unable to read files defined by the ST as system-wide credential stores.

- Test 41: The evaluator will attempt to read any other object specified in the assignment

Test not Applicable:	No other objects are specified in the assignment.

3.4.2 FPT_ASLR_EXT.1/Xeon, FPT_ASLR_EXT.1/z16, FPT_ASLR_EXT.1/Power10 Address Space Layout Randomization

3.4.2.1 Tests

80 The evaluator will select 3 executables included with the TSF. If the TSF includes a web browser it must be selected. If the TSF includes a mail client it must be selected. For each of these apps, the evaluator will launch the same executables on two separate instances of the OS on identical hardware and compare all memory mapping locations. The evaluator will ensure that no memory mappings are placed in the same location. If the rare chance occurs that two mappings are the same for a single executable and not the same for the other two, the evaluator will repeat the test with that executable to verify that in the second test the mappings are different. This test can also be completed on the same hardware and rebooting between application launches.

High-Level Test Description
Select a binary and execute it. Capture a snapshot of the process memory structure. Reboot. Select the same binary and execute it again. Capture the snapshot of the process memory structure. Compare the two memory structures and show that they are different.
Findings: PASS – The evaluator confirmed that for selected binaries they are subjected to ASLR support such that their memory maps are completely different on each invocation. The TOE does not include a web browser or email client.

81

3.4.3 FPT_SBOP_EXT.1 Stack Buffer Overflow Protection

3.4.3.1 Tests

82 For stack-based OSes, the evaluator will determine that the TSS contains a description of stack-based buffer overflow protections used by the OS. These are referred to by a variety of terms, such as stack cookie, stack guard, and stack canaries. The TSS must include a rationale for any binaries that are not protected in this manner.

Findings:	*This section includes a TSS activity.* Section 6.6.3 of the [ST] provides a description of the compiler option (-fstack-protector-strong) used in the TOE. This option is provided by the "gcc" compiler used

	for both the x86-64 and Z/Architecture builds to introduce stack guards for a wide range of stack-based functions. (Note that stack-protector-strong can protect more than stack overflows, but the scope of the analysis is on stack protection.) Within section 6.9 of the [ST], there are several binaries not compiled with stack protection. A reasonable rationale is provided for each category of file listed.

83 The evaluator will also perform the following test:

- Test 42: The evaluator will inventory the kernel, libraries, and application binaries to determine those that do not implement stack-based buffer overflow protections. This list should match up with the list provided in the TSS.

High-Level Test Description
The TOE is inventoried for kernel, libraries, and application binaries to determine those that do not implement stack-based buffer overflow protections. Ensure this list matches up with the list provided in the TSS.
Findings: PASS – The evaluator confirmed that the set of file comprising system executables, libraries, and kernel implement stack-based overflow protections consistent with the description in the [ST]. Any exceptions discovered were found to be consistent with the [ST].

84 For OSes that store parameters/variables separately from control flow values, the evaluator will verify that the TSS describes what data structures control values, parameters, and variables are stored. The evaluator will also ensure that the TSS includes a description of the safeguards that ensure parameters and variables do not intermix with control flow values.

Findings:	*This section includes a TSS activity.* The OS does not store parameters/variables separately from control flow values.

3.4.4 FPT_TST_EXT.1 Boot Integrity

3.4.4.1 TSS

85 The evaluator will verify that the TSS section of the ST includes a comprehensive description of the boot procedures, including a description of the entire bootchain, for the TSF. The evaluator will ensure that the OS cryptographically verifies each piece of software it loads in the bootchain to include bootloaders and the kernel. Software loaded for execution directly by the platform (e.g. first-stage bootloaders) is out of scope. For each additional category of executable code verified before execution, the evaluator will verify that the description in the TSS describes how that software is cryptographically verified.

Findings:	Section 6.6.5 of the [ST] describes the boot chain in comprehensive detail. While there is some information on the role the platform plays with respect to creating a boot chain of trust, the information denotes where the platform ends and the TOE begins from a security perspective. (Of particular note, an OS is not capable of verifying its own bootloader and this is why hardware/firmware-backed boot chain verification steps are needed – however, they are out of scope of the evaluation because the TOE does not actually perform the work.) The TSS claims that for the x86-64 platform, the grub2 bootloader – which is installed as part of the OS – is responsible for checking the kernel before it loads. The OS

	kernel is then responsible for verifying the signatures of each of the kernel drivers and modules. For the Z/Architecture, the Stage 3 Bootloader is where the TSF takes over from the platform. It is responsible for verifying the cryptographic signature on the kernel.

86 The evaluator will verify that the TSS contains a description of the protection afforded to the mechanism performing the cryptographic verification.

Findings:	The OS is one stage in a multi-stage booting process on both the x86-64 platform and the Z/Architecture system. The grub bootloader on the Dell and the Stage 3 Bootloader on the Z/Architecture are protected by parent chains operating in the firmware/hardware that ensure the OS bootloaders (essentially the first time the OS is capable of starting any of its own controlled code) are cryptographically verified. Since the entire boot chain of trust is not in scope (only the OS stages are in scope), it is necessary to discuss non-TOE components in the TSS as aiding the protection.

3.4.4.2 Tests

87 The evaluator will also perform the following tests:

- Test 43: The evaluator will perform actions to cause TSF software to load and observe that the integrity mechanism does not flag any executables as containing integrity errors and that the OS properly boots.

High-Level Test Description
Reboot the TOE. Show that the TOE boots without issues. Scan /var/log/messages to show that no integrity errors were encountered on boot.
Findings: PASS – The evaluator confirmed there are no errors found when the TOE is rebooted normally.

- Test 44: The evaluator will modify a TSF executable that is part of the bootchain verified by the TSF (i.e. Not the first-stage bootloader) and attempt to boot. The evaluator will ensure that an integrity violation is triggered and the OS does not boot (Care must be taken so that the integrity violation is determined to be the cause of the failure to load the module, and not the fact that in such a way to invalidate the structure of the module.).

High-Level Test Description
Modify the vmlinuz- boot image in the /boot directory. Reboot the TOE. Show that the reboot is unsuccessful.
Findings: PASS – The evaluator confirmed that a modified file in the boot chain will cause the TOE to generate an integrity error.

- Test 45[conditional, to be performed if
    - a digital signature using an X509 certificate with hardware-based protection is selected from FPT_TST_EXT.1.1

]: If the ST author indicates that the integrity verification is performed using public key in an X509 certificate, the evaluator will verify that the boot integrity mechanism includes a certificate validation according to FIA_X509_EXT.1 for all certificates in the chain from the certificate used for boot integrity to a certificate in the trust store that are not themselves in the trust store. This means that, for each X509 certificate in this chain that is not a trust store element, the evaluator must ensure that revocation information is available to the TOE during the bootstrap mechanism (before the TOE becomes fully operational).

Findings:	The TOE does not claim the use of X.509 certificates to protect boot integrity and therefore this test is N/A.

3.4.5 FPT_TUD_EXT.1 Trusted Update

3.4.5.1 Tests

88 The evaluator will check for an update using procedures described in the documentation and verify that the OS provides a list of available updates. Testing this capability may require installing and temporarily placing the system into a configuration in conflict with secure configuration guidance which specifies automatic update.

High-Level Test Description
Using `dnf` check for updates to any of the TOE OS files.
Findings: PASS – The evaluator confirmed that the TOE is capable of receiving a notification that updates are available for installed OS components.

89 The evaluator is also to ensure that the response to this query is authentic by using a digital signature scheme specified in FCS_COP.1/SIGN. The digital signature verification may be performed as part of a network protocol occurs over a trusted channel as described in FTP_ITC_EXT.1.) If the signature verification is not performed as part of a trusted channel, the evaluator will send a query response with a bad signature and verify that the signature verification fails. The evaluator will then send a query response with a good signature and verify that the signature verification is successful.

Findings:	*This section includes a TSS activity.* Section 6.6.6 of the [ST] specifies the user of RSA 4096 to verify the authenticity and integrity of updates. Section 6.8.1 of the [ST] specifies the use of TLS to protect checks for updates.

90 For the following tests, the evaluator will initiate the download of an update and capture the update prior to installation. The download could originate from the vendor's website, an enterprise-hosted update repository, or another system (e.g. network peer). All supported origins for the update must be indicated in the TSS and evaluated.

- Test 46: The evaluator will ensure that the update has a digital signature belonging to the vendor prior to its installation. The evaluator will modify the downloaded update in such a way that the digital signature is no longer valid. The evaluator will then attempt to install the modified update. The evaluator will ensure that the OS does not install the modified update.

High-Level Test Description
Verify the package has a digital signature using the rpm tool. Modify the update prior to installation. Attempt to install the update and show that the update fails to apply.
Findings: PASS – The evaluator confirmed that modifying the binary to invalidate the digital signature causes the TOE to reject installing the package.

- Test 47: The evaluator will ensure that the update has a digital signature belonging to the vendor. The evaluator will then attempt to install the update (or permit installation to continue). The evaluator will ensure that the OS successfully installs the update.

High-Level Test Description
Verify the package has a digital signature using the rpm tool. Attempt to install the update and show that the update applies successfully.
Findings: PASS – The evaluator confirmed that digitally signed updates can be installed successfully.

3.4.6 FPT_TUD_EXT.2 Trusted Update for Application Software

3.4.6.1 Tests

91 The evaluator will check for updates to application software using procedures described in the documentation and verify that the OS provides a list of available updates. Testing this capability may require temporarily placing the system into a configuration in conflict with secure configuration guidance which specifies automatic update.

Note:	The exact same mechanism for checking for updates to applications is tested in section 3.4.5.1.

92 The evaluator is also to ensure that the response to this query is authentic by using a digital signature scheme specified in FCS_COP.1/SIGN. The digital signature verification may be performed as part of a network protocol as described in FTP_ITC_EXT.1. If the signature verification is not performed as part of a trusted channel, the evaluator will send a query response with a bad signature and verify that the signature verification fails. The evaluator will then send a query response with a good signature and verify that the signature verification is successful.

Note:	The exact same mechanism for checking for updates to applications is tested in section 3.4.5.1.

93 The evaluator will initiate an update to an application. This may vary depending on the application, but it could be through the application vendor's website, a commercial app store, or another system. All origins supported by the OS must be indicated in the TSS and evaluated. However, this only includes those mechanisms for which the OS is providing a trusted installation and update functionality. It does not include user or administrator-driven download and installation of arbitrary files.

- Test 48: The evaluator will ensure that the update has a digital signature which chains to the OS vendor or another trusted root managed through the OS. The evaluator will modify the downloaded update in such a way that the digital signature is no longer valid. The evaluator will then attempt

to install the modified update. The evaluator will ensure that the OS does not install the modified update.

Note:	The exact same mechanism for checking for updates to applications is tested in section 3.4.5.1.

- Test 49: The evaluator will ensure that the update has a digital signature belonging to the OS vendor or another trusted root managed through the OS. The evaluator will then attempt to install the update. The evaluator will ensure that the OS successfully installs the update.

Note:	The exact same mechanism for checking for updates to applications is tested in section 3.4.5.1.

3.5 Audit Data Generation (FAU)

3.5.1 FAU_GEN.1 Audit Data Generation (Refined)

This section has been modified by TD 0693.

3.5.1.1 Guidance

94 The evaluator will check the administrative guide and ensure that it lists all of the auditable events. The evaluator will check to make sure that every audit event type selected in the ST is included.

Findings: This maps each audit event type included/selected in [ST] Section 5.3.1 FAU_GEN.1.1 to where it is listed in guidance:

- Start-up and shut-down of the audit functions – [AGD] Sections 10.2.1 and 10.2.2
- Authentication events (Success/Failure) – [AGD] Section 10.2.4
- Use of privileged/special rights events (Successful and unsuccessful security, audit, and configuration changes) – [AGD] Section 10.2.5
- Privilege or role escalation events (Success/Failure) – [AGD] Section 10.2.6
- File and object events (Successful and unsuccessful attempts to create, access, delete, modify, modify permissions) – [AGD] Section 10.2.7
- User and Group management events (Successful and unsuccessful add, delete, modify, disable, enable, and credential change) – [AGD] Section 10.2.8
- Audit and log data access events (Success/Failure) – [AGD] Section 10.2.9
- Cryptographic verification of software (Success/Failure) – [AGD] Section 10.2.10
- Attempted application invocation with arguments (Success/Failure e.g. due to software restriction policy) [AGD] Section 10.2.3
- System reboot, restart, and shutdown events (Success/Failure) – [AGD] Section 10.2.12
- Kernel module loading and unloading events (Success/Failure) – [AGD] Section 10.2.13
- Administrator or root-level access events (Success/Failure) – [AGD] Section 10.2.14
- Failure to establish SSH connection [AGD] Section 10.2.15.2
- Establishment of SSH connection [AGD] Section 10.2.15.1
- Termination of SSH connection session [AGD] Section 10.2.15.3
- Dropping of packet(s) outside of defined size limits [AGD] Section 10.2.15.4

95 The evaluator will check the administrative guide and ensure that it provides a format for audit records. Each audit record format type must be covered, along with a brief description of each field. The evaluator will ensure that the fields contains the information required.

Findings:	[AGD] Section 10.1 specifies a single format for audit records and briefly describes each field. While mapping the auditable events to the guidance in the previous Findings, the evaluator confirmed the fields contain the required information for each auditable event.

3.5.1.2 Tests

96 The evaluator will test the OS's ability to correctly generate audit records by having the TOE generate audit records for the events listed in the ST. This should include all instance types of an event specified. When verifying the test results, the evaluator will ensure the audit records generated during testing match the format specified in the administrative guide, and that the fields in each audit record provide the required information.

Findings:	Such audit record verification activities were performed as part of each of the test cases in which such audit messages were expected to be generated. The evaluator found that all audit messages were generated by the TOE and each audit message matched the format specified in the administrative guide.

3.6 Identification and Authentication (FIA)

3.6.1 FIA_AFL.1 Authentication failure handling (Refined)

This section has been modified by TD 0691.

3.6.1.1 Tests

97 The evaluator will set an administrator-configurable threshold for failed attempts, or note the ST-specified assignment. The evaluator will then (per selection) repeatedly attempt to authenticate with an incorrect password, PIN, or certificate until the number of attempts reaches the threshold. Note that the authentication attempts and lockouts must also be logged as specified in FAU_GEN.1.

- Test 53[conditional, to be performed if "authentication based on user name and password" is selected in FIA_AFL.1 and FIA_UAU.5]: The evaluator will attempt to authenticate repeatedly to the system with a known bad password. Once the defined number of failed authentication attempts has been reached the evaluator will ensure that the account that was being used for testing has had the actions detailed in the assignment list above applied to it. The evaluator will ensure that an event has been logged to the security event log detailing that the account has had these actions applied.

High-Level Test Description
Configure the number of attempts before lock out occurs. Using an unprivileged account, log in incorrectly just before the limit is reached to show that the account remains unlocked when a successful attempt is made. Then log in using incorrect credentials until the limit is reached and show that attempts to log in with valid credentials is rejected due to lock out.
Findings: PASS – The evaluator confirmed that when a user provides an invalid password more than the configured limit, the TOE locks the user account.

- Test 54[conditional, to be performed if "authentication based on user name and a PIN that releases an asymmetric key stored in OE-protected storage" is selected in FIA_AFL.1 and FIA_UAU.5]: The evaluator will attempt to authenticate repeatedly to the system with a known bad PIN. Once the defined number of failed authentication attempts has been reached the evaluator will ensure that the account that was being used for testing has had the actions detailed in the assignment list above applied to it. The evaluator will ensure that an event has been logged to the security event log detailing that the account has had these actions applied.

Findings:	The TOE does not claim the use of username and PIN as an authentication mechanism and therefore this test is N/A.

- Test 55[conditional, to be performed if "authentication based on X.509 certificates" is selected in FIA_AFL.1 and FIA_UAU.5]: The evaluator will attempt to authenticate repeatedly to the system using a known bad certificate. Once the defined number of failed authentication attempts has been reached the evaluator will ensure that the account that was being used for testing has had the actions detailed in the assignment list above applied to it. The evaluator will ensure that an event has been logged to the security event log detailing that the account has had these actions applied.

Findings:	The TOE does not claim the use of X.509 certificates as an authentication mechanism and therefore this test is N/A.

3.6.2 FIA_UAU.5 Multiple Authentication Mechanisms (Refined)

3.6.2.1 TSS

98 The evaluator will ensure that the TSS describes the rules as to how each authentication mechanism specified in FIA_UAU.5.1 is implemented and used. Example rules are how the authentication mechanism authenticates the user (i.e. how does the TSF verify that the correct password or authentication factor is used), the result of a successful authentication (i.e. is the user input used to derive or unlock a key) and which authentication mechanism can be used at which authentication factor interfaces (i.e. if there are times, for example, after a reboot, that only specific authentication mechanisms can be used). Rules regarding how the authentication factors interact in terms of unsuccessful authentication are covered in FIA_AFL.1.

Findings:	The TOE only claims locally provisioned usernames and passwords as well as the use of SSH public keys when authenticating using the SSH protocol. There are no claims for X.509 certificates or PINs.

	Section 6.4.2 provides the rules for ensuring that a user's entered password matches the password stored on the system. In addition, for the use of SSH user public keys, section 6.4.2 also indicates that the SSH_MSG_USERAUTH_REQUEST is a signed message which must be verified by the TOE's SSH server when authenticating users. There are no rules cited as to when certain authentication mechanisms are available.

3.6.2.2 Guidance

99 The evaluator will verify that configuration guidance for each authentication mechanism is addressed in the AGD guidance.

Findings:	[AGD] Section 6.3 and [RHEL] section 9.2.1 describe managing user passwords. [AGD] section 5.1.1.1 describes configuring SSH public key authentication.

3.6.2.3 Tests

100 The following content should be included if:

- authentication based on username and password is selected from FIA_UAU.5.1

  - Test 56: The evaluator will attempt to authenticate to the OS using the known user name and password. The evaluator will ensure that the authentication attempt is successful.

High-Level Test Description
Using both the local console and remote console, attempt to log into the TOE using a known-good username and password. Show that the attempt is successful and that an auditable event is generated.
Findings: PASS – The evaluator confirmed that defined users provisioned with passwords can log in successfully to the local console and remote console when the proper password is provided.

  - Test 57: The evaluator will attempt to authenticate to the OS using the known user name but an incorrect password. The evaluator will ensure that the authentication attempt is unsuccessful.

High-Level Test Description
Using both the local console and remote console, attempt to log into the TOE using a known-good username and an incorrect password. Show that the attempt is unsuccessful and that an auditable event is generated.
Findings: PASS – The evaluator confirmed that defined users provisioned with passwords fail to log in to the local console and remote console when an incorrect password is provided.

101 The following content should be included if:

- username and a PIN that releases an asymmetric key is selected from FIA_UAU.5.1

The evaluator will examine the TSS for guidance on supported protected storage and will then configure the TOE or OE to establish a PIN which enables release of the asymmetric key from the protected storage (such as a TPM, a hardware token, or isolated execution environment) with which the OS can interface.

Findings:	*This section includes a TSS and/or guidance activity.* The TOE does not claim username and PIN.

The evaluator will then conduct the following tests:

- Test 58: The evaluator will attempt to authenticate to the OS using the known user name and PIN. The evaluator will ensure that the authentication attempt is successful.

Findings:	The TOE does not claim username and PIN and therefore this test is N/A.

- Test 59: The evaluator will attempt to authenticate to the OS using the known user name but an incorrect PIN. The evaluator will ensure that the authentication attempt is unsuccessful.

Findings:	The TOE does not claim username and PIN and therefore this test is N/A.

102 The following content should be included if:

- combination of authentication based on user name, password, and time-based one-time password is selected from FIA_UAU.5.1

  The evaluator will configure the OS to authentication to authenticate to the OS using a username, password, and one-time password mechanism. The evaluator will then perform the following tests.

  - Test 60: The evaluator will attempt to authenticate using a valid username, valid password, and valid one-time password. The evaluator will ensure that the authentication attempt is successful.

Findings:	The TOE does not claim a combination of username, password and one-time passwords and therefore this test is N/A.

  - Test 61: The evaluator will attempt to authenticate using a valid username, invalid password, and valid one-time password. The evaluator will ensure that the authentication attempt fails.

Findings:	The TOE does not claim a combination of username, password and one-time passwords and therefore this test is N/A.

  - Test 62: The evaluator will attempt to authenticate using a valid username, valid password, and invalid one-time password. The evaluator will ensure that the authentication attempt fails.

Findings:	The TOE does not claim a combination of username, password and one-time passwords and therefore this test is N/A.

- Test 63: The evaluator will attempt to authenticate using a valid username, invalid password, and invalid one-time password. The evaluator will ensure that the authentication attempt fails.

Findings:	The TOE does not claim a combination of username, password and one-time passwords and therefore this test is N/A.

103 Authentication mechanisms related to authentication based on X.509 certificates are tested under FIA_X509_EXT.1 and SSH public key-based authentication are tested in the Functional Package for Secure Shell (SSH), version 1.0.

104 For each authentication mechanism rule, the evaluator will ensure that the authentication mechanism(s) behave as documented in the TSS.

Findings:	As found in the testing described above, the authentication mechanisms were found to behave as per the description found in the TSS.

3.6.3 FIA_X509_EXT.1 X.509 Certificate Validation

	This section has been modified by TD 0773.

3.6.3.1 TSS

105 The evaluator will ensure the TSS describes where the check of validity of the certificates takes place. The evaluator ensures the TSS also provides a description of the certificate path validation algorithm.

Findings:	Section 6.4.3 of the [ST] specifies that certification validation occurs when authenticating TLS and HTTPS connections. Specifically, when the TOE client receives the certificate from the remote endpoint during the TLS handshake. The certificate path validation algorithm is provided as a series of high-level bullet-points in section 6.4.3.

106 If there are exceptional use cases where the OS cannot perform revocation checking in accordance with at least one of the revocation methods, the evaluator will ensure the TSS describes each revocation checking exception use case and, for each exception, the alternate functionality the TOE implements to determine the status of the certificate and disable functionality dependent on the validity of the certificate.

Findings:	Section 6.4.3 of the [ST] indicates the certificate is rejected if the validity check fails.

3.6.3.2 Tests

107 The tests described must be performed in conjunction with the other certificate services evaluation activities, including the functions in FIA_X509_EXT.2.1. The evaluator will create a chain of at least four certificates: the node certificate to be tested, two Intermediate CAs, and the self-signed Root CA.

- Test 64: The evaluator will demonstrate that validating a certificate without a valid certification path results in the function failing, for each of the following reasons, in turn: by establishing a certificate path in which one of the issuing certificates is not a CA certificate, by omitting the basicConstraints field in one of the issuing certificates, by setting the basicConstraints field in an issuing certificate to have CA=False, by omitting the CA signing bit of the key usage field in an issuing certificate, and by setting the path length field of a valid CA field to a value strictly less than the certificate path. The evaluator will then establish a valid certificate path consisting of valid CA certificates, and demonstrate that the function succeeds. The evaluator will then remove trust in one of the CA certificates, and show that the function fails.

High-Level Test Description
Show the TOE successfully establishes a connection when a valid cert chain is presented. Show the TOE does not establish a conneciton when the cert chain is invalid for one of the following reasons: - one of the issuing certificates is not a CA certificate, - one of the issuing certificates does not contain the basicConstraints extension, - one of the issuing certificates contains the basicConstraints extension with CA=False, - one of the issuing certificates does not contain the CA signing bit, - one of the issuing certificates has a path length field less than the certificate path, - the certificate chain is incomplete (i.e., missing a intermediate CA certificate)
Findings: PASS – The evaluator confirmed that the following invalid certificates cause the TOE to reject the certificate chain in the TLS connection: - a certificate path in which one of the issuing certificates is not a CA certificate, - one of the issuing certificates does not contain the basicConstraints extension, - one of the issuing certificates contains the basicConstraints extension with CA=False, - by omitting the CA signing bit of the key usage field in an issuing certificate, - by setting the path length field of a valid CA field to a value strictly less than the certificate path, - the certificate chain is incomplete (i.e., missing a intermediate CA certificate).

- Test 65: The evaluator will demonstrate that validating an expired certificate results in the function failing.

High-Level Test Description
Using a TOE TLS client, connect to a TLS server which will return back an expired certificate and show the connection fails.
Findings: PASS – The evaluator confirmed that an expired certificate prevents successful validation of the certificate chain.

- Test 66: [Conditional, to be performed for use cases identified in exceptions that cannot be configured to allow revocation checking] The evaluator will test that the OS can properly handle revoked certificates - conditional on whether CRL, OCSP, OCSP stapling, or OCSP multi-stapling is selected; if multiple methods are selected, then a test will be performed for each method. The evaluator will test revocation of the node certificate and revocation of the intermediate CA certificate (i.e. the intermediate CA certificate should be revoked by the root CA). If OCSP stapling per RFC 6066 is the only supported revocation method,

testing revocation of the intermediate CA certificate is omitted. The evaluator will ensure that a valid certificate is used, and that the validation function succeeds. The evaluator then attempts the test with a certificate that has been revoked (for each method chosen in the selection) to ensure when the certificate is no longer valid that the validation function fails. If the exceptions are configurable, the evaluator shall attempt to configure the exceptions to allow revocation checking for each function indicated in FIA_X509_EXT.2.

High-Level Test Description
Using the TLS client provided by the TOE, make a connection to a TLS server in the TOE while sending Certificate Status Request extension. Show that when a certificate is flagged as revoked in the stapled OCSP response, the connection fails.
Findings: PASS – The evaluator confirmed that using OCSP stapling results in the TOE TLS component being able to reject the certificate due to revocation.

- Test 67: If any OCSP option is selected, the evaluator will configure the OCSP server or use a man-in-the-middle tool to present a certificate that does not have the OCSP signing purpose and verify that validation of the OCSP response fails. If CRL is selected, the evaluator will configure the CA to sign a CRL with a certificate that does not have the cRLsign key usage bit set and verify that validation of the CRL fails.

High-Level Test Description
Using a TLS client, connect to the TLS server and verify that the TLS client will fail to validate the OCSP response when the response is signed by a CA which does not have the proper policy flag extension set.
Findings: PASS – The evaluator confirmed that processing an OCSP response signed by a certificate which is missing the OCSP Signing purpose results in the TLS application failing to validate the OCSP response and hence the certificate chain.

- Test 68: The evaluator will modify any byte in the first eight bytes of the certificate and demonstrate that the certificate fails to validate. (The certificate will fail to parse correctly.)

High-Level Test Description
Force the TOE to connect to a Lightship test server which will send back a properly mangled X.509 certificate in which the ASN.1 header bytes in the first 8 bytes are modified.
Findings: PASS – The evaluator confirmed that the TOE will reject a certificate in which a byte in the first eight bytes has been modified.

- Test 69: The evaluator will modify any byte in the last eight bytes of the certificate and demonstrate that the certificate fails to validate. (The signature on the certificate will not validate.)

High-Level Test Description
Force the TOE to connect to a Lightship test server which will send back an X.509 certificate in which the last byte of the certificate (the signature) is modified.
Findings: PASS – The evaluator confirmed that the TOE will reject a certificate in which a byte in the digital signature has been modified.

- Test 70: The evaluator will modify any byte in the public key of the certificate and demonstrate that the certificate fails to validate. (The signature of the certificate will not validate.)

High-Level Test Description
Force the TOE to connect to a Lightship test server which will send back an X.509 certificate in which the public key of the certificate is modified.
Findings: PASS – The evaluator confirmed that the TOE will reject a certificate in which a byte in the public key has been modified.

- Test 71[conditional, to be performed if
    - ECDSA schemes is selected from FCS_COP.1.1/SIGN
    - 6187 is selected from FCS_SSH_EXT.1.1 from Functional Package for Secure Shell (SSH), version 1.0

  ]:

    - Test 71.1: The evaluator will establish a valid, trusted certificate chain consisting of an EC leaf certificate, an EC Intermediate CA certificate not designated as a trust anchor, and an EC certificate designated as a trusted anchor, where the elliptic curve parameters are specified as a named curve. The evaluator will confirm that the TOE validates the certificate chain.

High-Level Test Description
Construct a chain of ECDSA certificates using named curves. Show that the leaf and chain using named curves can be used to establish a successful connection.
Findings: PASS – The evaluator confirmed that using an ECDSA certificate chain in which all certificates use a named P-curve results in the TOE successfully validating the certificate chain.

    - Test 71.2: The evaluator will replace the intermediate certificate in the certificate chain for Test 71.1 with a modified certificate, where the modified intermediate CA has a public key information field where the EC parameters uses an explicit format version of the Elliptic Curve parameters in the public key information field of the intermediate CA certificate from Test 71.1, and the modified Intermediate CA certificate is signed by the trusted EC root CA, but having no other changes. The evaluator will confirm the TOE treats the certificate as invalid.

High-Level Test Description
Construct a chain of ECDSA certificates. Create a clone of the Intermediate CA, such that the public key is explicitly defined rather than being a named curve. Show that the leaf and chain are not validated correctly when connected to.
Findings: PASS – The evaluator confirmed that using an ECDSA certificate chain in which the intermediate certificate uses an explicitly parameterized version of an otherwise valid named P-curve results in the TOE failing to validate the certificate chain.

- Test 72[conditional, to be performed if
    - exceptional use cases is selected from FIA_X509_EXT.1.1

  ]: For each exceptional use case for revocation checking described in the ST, the evaluator shall attempt to establish the conditions of the use case, designate the certificate as invalid and perform the function relying on the certificate. The evaluator shall observe that the alternate revocation checking mechanism successfully prevents performance of the function.

Findings:	The TOE does not claim exceptions to revocation checking and therefore this test is N/A.

108 [Conditional, to be performed if "authentication based on X.509 certificates" is selected in FIA_UAU.5]: The evaluator will generate an X.509v3 certificate for a user with the Client Authentication Extended Key Usage field set. The evaluator will provision the OS for authentication with the X.509v3 certificate. The evaluator will ensure that the certificates are validated by the OS as per FIA_X509_EXT.1.1 and then conduct the following two tests:

- Test 73: The evaluator will attempt to authenticate to the OS using the X.509v3 certificate. The evaluator will ensure that the authentication attempt is successful.

Findings:	The TOE does not claim “authentication based on X.509 certificates” and therefore this test is N/A.

- Test 74: The evaluator will generate a second certificate identical to the first except for the public key and any values derived from the public key. The evaluator will attempt to authenticate to the OS with this certificate. The evaluator will ensure that the authentication attempt is unsuccessful.

Findings:	The TOE does not claim “authentication based on X.509 certificates” and therefore this test is N/A.

109 The tests described must be performed in conjunction with the other certificate services evaluation activities, including the functions in FIA_X509_EXT.2.1. The evaluator will create a chain of at least four certificates: the node certificate to be tested, two Intermediate CAs, and the self-signed Root CA.

- Test 75: The evaluator will construct a certificate path, such that the certificate of the CA issuing the OS's certificate does not contain the basicConstraints extension. The validation of the certificate path fails.

Note:	This is performed as part of FIA_X509_EXT.1 test 64.

- Test 76: The evaluator will construct a certificate path, such that the certificate of the CA issuing the OS's certificate has the CA flag in the basicConstraints extension not set. The validation of the certificate path fails.

Note:	This is performed as part of FIA_X509_EXT.1 test 64.

- Test 77: The evaluator will construct a certificate path, such that the certificate of the CA issuing the OS's certificate has the CA flag in the basicConstraints extension set to TRUE. The validation of the certificate path succeeds.

High-Level Test Description
Using a TLS client, connect to a TLS server such that the leaf certificate issuing Intermediate CA has a basicConstraint extension in which the CA flag is set to an explicit non-zero value representing the logical value of TRUE.
Findings: PASS – The evaluator confirmed that when the Intermediate certificate has a basicConstraints extension CA flag set to True (a non-zero value in the ASN.1 encoding), the TOE successfully validates the certificate chain.

3.6.4 FIA_X509_EXT.2 X.509 Certificate Authentication

This section has been modified by TD 0789.

3.6.4.1 Tests

110 The evaluator will acquire or develop an application that uses the selected OS mechanism with an X.509v3 certificate. The evaluator will then run the application and ensure that the provided certificate is used to authenticate the connection.

111 The evaluator will repeat the activity for all selections listed.

Note:	FIA_X509_EXT.1 testing was performed using the 'dnf' program which uses the OS TLS and HTTPS mechanisms with an X.509 certificate. The FIA_X509_EXT.1 tests show the X.509 certificate is used to authenticate the connection.

3.7 Trusted path/channels (FTP)

3.7.1 FTP_ITC_EXT.1 Trusted channel communication

This section has been modified by TD 0789.

3.7.1.1 Tests

112 The evaluator shall configure the OS to communicate with another trusted IT product as identified in the third selection. The evaluator shall monitor network traffic while the OS performs communication with each of the servers identified in the third selection. The evaluator shall ensure that for each session a trusted channel was established in conformance with the selected protocols.

Findings:	FTP_ITC_EXT.1 identifies communicating with servers for Application initiated TLS, software updates, and connections to remote SSH servers. FCS_TLSC_EXT tests show the Application initiated TLS connection was established in conformance with TLS client requirements. Software updates (dnf) invoke Application initiated TLS. FCS_SSH_EXT and FCS_SSHC_EXT tests show the connections to remote SSH servers are conformant with the SSH client requirements.

3.7.2 FTP_TRP.1 Trusted Path

3.7.2.1 TSS

113 The evaluator will examine the TSS to determine that the methods of remote OS administration are indicated, along with how those communications are protected. The evaluator will also confirm that all protocols listed in the TSS in support of OS administration are consistent with those specified in the requirement, and are included in the requirements in the ST.

Findings:	The [ST] in section 6.8.2 claims the use of SSH as the only method for remote administration. The SSH protocol protects the administrative session. The evaluator confirmed that SSH is claimed in section 5.3.2 of the [ST] and is consistent with the claims in FTP_TRP.1 in section 5.3.8 of the [ST].

3.7.2.2 Guidance

114 The evaluator will confirm that the operational guidance contains instructions for establishing the remote administrative sessions for each supported method.

Findings:	[AGD] Section 6.13 provides guidance for establishing remote administrative sessions using SSH.

3.7.2.3 Tests

115 The evaluator will also perform the following tests:

- Test 78: The evaluator will ensure that communications using each remote administration method is tested during the course of the evaluation, setting up the connections as described in the operational guidance and ensuring that communication is successful.

Findings:	The only remote administration method is via SSH. The SSH protocol is successfully set up, and tested throughout the other test cases, especially in FCS_SSH_EXT.1 and FCS_SSHS_EXT.1.

- Test 79: For each method of remote administration supported, the evaluator will follow the operational guidance to ensure that there is no available interface that can be used by a remote user to establish a remote administrative sessions without invoking the trusted path.

High-Level Test Description
Perform a port scan of the device and determine if there are any remote administrative interfaces available outside of the SSH CLI interface.
Findings: PASS – The evaluator confirmed there are no additional remote management interfaces offered by the TOE other than those identified.

- Test 80: The evaluator will ensure, for each method of remote administration, the channel data is not sent in plaintext.

High-Level Test Description
Using a packet sniffer, show that the channel data is not being sent in plaintext for the SSH CLI trusted path.
Findings: PASS – The evaluator confirmed that use of the trusted path results in no data being transmitted in the clear.

- Test 81: The evaluator will ensure, for each method of remote administration, modification of the channel data is detected by the OS.

High-Level Test Description
Using a custom tool, modify SSH application data traffic originating from the non-TOE endpoint and show that the modification is detected by the TOE.
Findings: PASS – The evaluator confirmed that the TOE will successfully detect the modification of traffic sent over the trusted path.

4 Strictly Optional Requirements

This section has been modified by TD 0675 to move FPT_W^X_EXT.1 as strictly optional.

4.1 TOE Access (FTA)

4.1.1 FTA_TAB.1 Default TOE access banners

4.1.1.1 Tests

116 The evaluator will configure the OS, per instructions in the OS manual, to display the advisory warning message "TEST TEST Warning Message TEST TEST". The evaluator will then log out and confirm that the advisory message is displayed before logging in can occur.

High-Level Test Description
Configure the appropriate TSFI files for each of the defined interfaces. Set the contents of the file to "TEST TEST Warning Message TEST TEST" (along with another indicator to show correct file usage). Log into the defined management interfaces and show that the TOE banner is displayed prior to a completed login session.
Findings: PASS – The evaluator confirmed that the TOE is capable of displayed an advisory message prior to a successful login session.

5 Objective Requirements

5.1.1 FPT_SRP_EXT.1 Software Restriction Policies

5.1.1.1 TSS

117 The evaluator will ensure that the description of the supported characteristics in the TSS is consistent with the SFR. The evaluator will also ensure that any characteristics specified by the ST-author are described in sufficient detail to understand how to test those characteristics.

Findings:	Section 6.6.4 of the [ST] claims the use of "file path" and "hash" as the characteristics. The evaluator found these to be consistent with the claims made in section 5.3.6 of the [ST]. The use of the file path and the hash are self-explanatory and can be easily mapped to test cases found in the [OSPP] test EAs.

5.1.1.2 Guidance

118 The evaluator will ensure that that the characteristics are described in sufficient detail for administrators to configure policies using them, and that the list of characteristics in the guidance is consistent with the information in the TSS.

Findings:	[AGD] section 5.4 describes the configuration of software restriction policies. The described policies are consistent with the TSS.

5.1.1.3 Tests

119 There are two tests for each selection above.

- Test 84[conditional, to be performed if
  - file path is selected from FPT_SRP_EXT.1.1

  ]: The evaluator will configure the OS to only allow code execution from the core OS directories. The evaluator will then attempt to execute code from a directory that is in the allowed list. The evaluator will ensure that the code they attempted to execute has been executed.

High-Level Test Description
Run a binary from /bin and show that the binary is permitted.
Findings: PASS – The evaluator confirmed that applications that are considered trusted – which includes binaries installed as part of the package management in /bin – are permitted to be executed.

- Test 85[conditional, to be performed if
  - file path is selected from FPT_SRP_EXT.1.1

  ]: The evaluator will configure the OS to only allow code execution from the core OS directories. The evaluator will then attempt to execute code from a directory that is not in the allowed list. The evaluator will ensure that the code they attempted to execute has not been executed.

High-Level Test Description
Run a binary from /tmp and show that the binary is not permitted to be executed.
Findings: PASS – The evaluator confirmed that running an application in a directory not considered trusted are not permitted to run.

- Test 86[conditional, to be performed if
  - file digital signature is selected from FPT_SRP_EXT.1.1

  ]: The evaluator will configure the OS to only allow code that has been signed by the OS vendor to execute. The evaluator will then attempt to execute code signed by the OS vendor. The evaluator will ensure that the code they attempted to execute has been executed.

Findings:	The TOE does not claim “file digital signature” and therefore this test is N/A.

- Test 87[conditional, to be performed if
  - file digital signature is selected from FPT_SRP_EXT.1.1

  ]: The evaluator will configure the OS to only allow code that has been signed by the OS vendor to execute. The evaluator will then attempt to execute code signed by another digital authority. The evaluator will ensure that the code they attempted to execute has not been executed.

Findings:	The TOE does not claim “file digital signature” and therefore this test is N/A.

- Test 88[conditional, to be performed if
  - version is selected from FPT_SRP_EXT.1.1

  ]: The evaluator will configure the OS to allow execution of a specific application based on version. The evaluator will then attempt to execute the same version of the application. The evaluator will ensure that the code they attempted to execute has been executed.

Findings:	The TOE does not claim “version” and therefore this test is N/A.

- Test 89[conditional, to be performed if
  - version is selected from FPT_SRP_EXT.1.1

  ]: The evaluator will configure the OS to allow execution of a specific application based on version. The evaluator will then attempt to execute an older version of the application. The evaluator will ensure that the code they attempted to execute has not been executed.

Findings:	The TOE does not claim “version” and therefore this test is N/A.

- Test 90[conditional, to be performed if
  - hash is selected from FPT_SRP_EXT.1.1

]: The evaluator will configure the OS to allow execution based on the hash of the application executable. The evaluator will then attempt to execute the application with the matching hash. The evaluator will ensure that the code they attempted to execute has been executed.

High-Level Test Description
Run a binary from /tmp – which was previously blocked – but ensure it is permitted by hash, and show that the binary is permitted.
Findings: PASS – The evaluator confirmed that running an application which is allowed by hash is permitted.

- Test 91[conditional, to be performed if
  - hash is selected from FPT_SRP_EXT.1.1

  ]: The evaluator will configure the OS to allow execution based on the hash of the application executable. The evaluator will modify the application in such a way that the application hash is changed. The evaluator will then attempt to execute the application with the matching hash. The evaluator will ensure that the code they attempted to execute has not been executed.

High-Level Test Description
Modify a previously permitted binary to no longer match the permitted hash. Then try to execute the binary. Show that the binary fails to run.
Findings: PASS – The evaluator confirmed that running an application which is allowed by hash fails to run if the hash does not match the permitted hashes in the ruleset.

- Test 92[conditional, to be performed if
  - other is selected from FPT_SRP_EXT.1.1

  ]: The evaluator will attempt to run an application that should be allowed based on the defined software restriction policy and ensure that it runs.

Findings:	The TOE does not claim “other” mechanisms and therefore this test is N/A.

- Test 93[conditional, to be performed if
  - other is selected from FPT_SRP_EXT.1.1

  ]: The evaluator will then attempt to run an application that should not be allowed the defined software restriction policy and ensure that it does not run.

Findings:	The TOE does not claim “other” mechanisms and therefore this test is N/A.

6 Implementation-based Requirements

120 This evaluation does not claim any Implementation-based requirements.

7 Additional Packages

7.1 Cryptographic Support (FCS)

7.1.1 FCS_TLS_EXT.1 TLS Protocol

7.1.1.1 FCS_TLS_EXT.1 Guidance Documentation

121 The evaluator shall ensure that the selections indicated in the ST are consistent with selections in the dependent components.

Findings:	[ST] Section 5.3.2 FCS_TLS_EXT.1.1 selects "TLS as a client." The evaluator confirmed this is consistent with [ST] section 5.3.2 including FCS_TLSC_EXT.* SFRs and no TLS server or DTLS SFRs.

7.1.2 FCS_TLSC_EXT.1 TLS Client Protocol

This section has been modified by TD 0499 and TD 0513.

7.1.2.1 FCS_TLSC_EXT.1.1 TSS

122 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.

Findings:	The [ST] section 6.2.12 claims support for specific ciphersuites. These ciphersuites are consistent with the allowable set of ciphersuites permitted in the SFR. The evaluator confirmed that the ciphersuites in section 6.2.12 of the [ST] are consistent with the selections in section 5.3.2 of the [ST].

7.1.2.2 FCS_TLSC_EXT.1.1 Guidance Documentation

123 The evaluator shall also check the operational guidance to ensure that it contains instructions on configuring the product so that TLS conforms to the description in the TSS.

Findings:	[AGD] Section 6.10 provides instructions for enforcing the use of TLSv1.2 and indicates the ciphersuites are enforced via the ospp SCAP profile.

7.1.2.3 FCS_TLSC_EXT.1.1 Tests

124 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).

High-Level Test Description
Using a TLS client, connect to the TLS server in the environment and show that the claimed ciphersuites are supported.
Findings: PASS – The evaluator confirmed that the TOE TLS client is capable of successfully negotiating the claimed ciphersuites.

- **Test 2**: The goal of the following test is to verify that the TOE accepts only certificates with appropriate values in the extendedKeyUsage extension, and implicitly that the TOE correctly parses the extendedKeyUsage extension as part of X.509v3 server certificate validation.

  The evaluator shall attempt to establish the connection using a server with a server certificate that contains the Server Authentication purpose in the extendedKeyUsage extension and verify that a connection is established. The evaluator shall repeat this test using a different, but otherwise valid and trusted, certificate that lacks the Server Authentication purpose in the extendedKeyUsage extension and ensure that a connection is not established. Ideally, the two certificates should be similar in structure, the types of identifiers used, and the chain of trust.

High-Level Test Description
Using the TOE TLS client, show that when connecting to the TLS server in the environment which is missing the serverAuth property in the extendedKeyUsage extension, that the TLS client will terminate.
Findings: PASS – The evaluator confirmed the TOE TLS library is capable of rejecting X.509 certificates where the extendedKeyUsage is missing the serverAuth policy.

- **Test 3**: The evaluator shall send a server certificate in the TLS connection that does not match the server-selected cipher suite (for example, send a ECDSA certificate while using the TLS_RSA_WITH_AES_128_CBC_SHA cipher suite or send a RSA certificate while using one of the ECDSA cipher suites.) The evaluator shall verify that the product disconnects after receiving the server's Certificate handshake message.

High-Level Test Description
Using a TOE TLS client, connect to a TLS server in the environment which sends back a certificate which does not match the negotiated ciphersuite and show that the connection is terminated.
Findings: PASS – The evaluator confirmed that when there is a mismatch between the certificate type and the ciphersuite, the TOE TLS client will terminate with an error.

- **Test 4**: The evaluator shall configure the server to select the TLS_NULL_WITH_NULL_NULL cipher suite and verify that the client denies the connection.

High-Level Test Description
Using a TLS client, connect to the TLS server in the environment using the TLS_NULL_WITH_NULL_NULL cipher and show that the connection is denied.

High-Level Test Description
Findings: PASS – The evaluator confirmed that the TOE TLS client will fail to connect when there is a ciphersuite mismatch.

- **Test 5**: The evaluator shall perform the following modifications to the traffic:

    1. **Test 5.1**: Change the TLS version selected by the server in the Server Hello to an undefined TLS version (for example 1.5 represented by the two bytes 03 06) and verify that the client rejects the connection.

High-Level Test Description
Connect to a TLS server in the environment and show that the TOE TLS client fails to connect when presented with an incorrect protocol version string.
Findings: PASS – The evaluator confirmed that the TOE TLS client will fail to connect when there is a protocol version mismatch.

    2. **Test 5.2**: Change the TLS version selected by the server in the Server Hello to the most recent unsupported TLS version (for example 1.1 represented by the two bytes 03 02) and verify that the client rejects the connection.

High-Level Test Description
Connect to a TLS server in the environment and show that the TOE TLS client fails to connect when presented with an incorrect protocol version string.
Findings: PASS – The evaluator confirmed that the TOE TLS client will fail to connect when there is a protocol version mismatch.

    3. **Test 5.3**: [conditional] If DHE or ECDHE cipher suites are supported, modify at least one byte in the server's nonce in the Server Hello handshake message, and verify that the client does not complete the handshake and no application data flows.

High-Level Test Description
Using the TOE TLS client, connect to the TLS server in the environment and show that the client rejects the connection when the server key exchange message has been modified as per the test case.
Findings: PASS – The evaluator confirmed that the TOE TLS client will fail to connect when the initial Server Hello nonce is modified.

    4. **Test 5.4**: Modify the server's selected cipher suite in the Server Hello handshake message to be a cipher suite not presented in the Client Hello handshake message. The evaluator shall verify that the client does not complete the handshake and no application data flows.

High-Level Test Description
Using a TLS client, connect to the TLS server in the environment which will attempt to negotiate a ciphersuite not presented in the TOE's Client Hello ciphersuite list. Show the connection is denied.

High-Level Test Description
Findings: PASS – The evaluator confirmed that the TOE TLS client will fail to connect when the server returns ciphersuites not supported by the initial TLS Client Hello.

5. **Test 5.5**: [conditional] If DHE or ECDHE cipher suites are supported, modify the signature block in the server's Key Exchange handshake message, and verify that the client does not complete the handshake and no application data flows. This test does not apply to cipher suites using RSA key exchange. If a TOE only supports RSA key exchange in conjunction with TLS, then this test shall be omitted.

High-Level Test Description
Using the TOE TLS client, connect to a TLS server in the environment which will modify the Server Key Exchange handshaking message signature and verify the connection fails.
Findings: PASS – The evaluator confirmed that the TOE TLS client will fail to connect when the server returns a mangled Server Key Exchange signature block.

6. **Test 5.6**: Modify a byte in the Server Finished handshake message, and verify that the client does not complete the handshake and no application data flows.

High-Level Test Description
Using the TOE TLS client, connect to a TLS server in the environment which will modify the Finished handshaking message and verify the connection does not send any Application Data.
Findings: PASS – The evaluator confirmed that the TOE TLS client will issue a Fatal Alert message and fail to connect when the server returns a mangled Server Finished message.

7. **Test 5.7**: Send a message consisting of random bytes from the server after the server has issued the Change Cipher Spec message and verify that the client does not complete the handshake and no application data flows. The message must still have a valid 5-byte record header in order to ensure the message will be parsed as TLS.

High-Level Test Description
Using the TOE TLS client, connect to a TLS server in the environment which will send back a garbled message after the ChangeCipherSpec has been issued and show the connection fails.
Findings: PASS – The evaluator confirmed that the TOE TLS client will fail to connect when the server returns a final Encrypted Handshake message which cannot be properly decrypted by the client.

7.1.2.4 FCS_TLSC_EXT.1.2 TSS

125 The evaluator shall ensure that the TSS describes the client's method of establishing all reference identifiers from the application-configured reference identifier, including which types of reference identifiers are supported (e.g. Common Name, DNS Name, URI Name, Service Name, or other application-specific Subject Alternative Names) and whether IP addresses and wildcards are supported. The evaluator shall ensure that this description identifies whether and the manner in which certificate pinning is supported or used by the product.

Findings:	Section 6.2.12 of the [ST] claims that the TOE establishes the reference identifier by parsing the DNS Name or IP address for the configured TLS server. A specific TLS client instance employed by the TOE is the TLS client used to check for updates to the TOE and its applications as part of FPT_TUD_EXT.1 and FPT_TUD_EXT.2. The TLS client will parse the DNS name or IP reference identifier from the URL. The evaluator noted that section 6.2.12 of the [ST] explicitly indicates no support for SRV or URI identifier types. Instead, the TOE supports use of IP addresses and DNS names. DNS names can be in the Common Name of the X.509 certificate or in the DNS Subject Alternative Name (SAN) extension. IP addresses are only permitted to be located in the IP SAN. The evaluator confirmed that the TSS in section 6.2.12 of the [ST] claims that wildcards are supported for DNS names only. The evaluated also noted that section 6.2.12 of the [ST] explicitly claims no support of certificate pinning.

7.1.2.5 FCS_TLSC_EXT.1.2 Guidance Documentation

126 The evaluator shall verify that the AGD guidance includes instructions for setting the reference identifier to be used for the purposes of certificate validation in TLS.

Findings:	[AGD] Section 6.10 indicates the reference identifier can be configured using the CURLOPT_URL and URLOPT_SSL_VERIFYHOST parameters.

7.1.2.6 FCS_TLSC_EXT.1.2 Tests

127 The evaluator shall configure the reference identifier according to the AGD guidance and perform the following tests during a TLS connection. If the TOE supports certificate pinning, all pinned certificates must be removed before performing Tests 1 through 6. A pinned certificate must be added prior to performing Test 7.

- **Test 1**: The evaluator shall present a server certificate that contains a CN that does not match the reference identifier and does not contain the SAN extension. The evaluator shall verify that the connection fails.

  Note that some systems might require the presence of the SAN extension. In this case the connection would still fail but for the reason of the missing SAN extension instead of the mismatch of CN and reference identifier. Both reasons are acceptable to pass Test 1.

High-Level Test Description
Using the TOE TLS client, connect to a TLS server in the environment which will send back a certificate in which the Common Name does not match the reference name and does not contain the SAN extension. Show the connection will fail. Repeat for IPv4 and hostname reference names. Ensure the test case is perform for each of the claimed identifier types.
Findings: PASS – The evaluator confirmed that certificates which contain invalid CN are rejected by the TOE TLS client.

- **Test 2**: The evaluator shall present a server certificate that contains a CN that matches the reference identifier, contains the SAN extension, but does not contain an identifier in the SAN that matches the reference identifier. The evaluator shall verify that the connection fails. The evaluator shall repeat this test for each supported SAN type.

High-Level Test Description
Using the TOE TLS client, connect to the TLS server in the environment. The server will return an X.509 certificate which has a CN that matches and a SAN which does not. The TLS client is expected to fail to connect. Repeat for each identifier type.
Findings: PASS – The evaluator confirmed that certificates which contain valid CN but invalid SAN values are rejected by the TOE TLS client.

- **Test 3**: [conditional] If the TOE does not mandate the presence of the SAN extension, the evaluator shall present a server certificate that contains a CN that matches the reference identifier and does not contain the SAN extension. The evaluator shall verify that the connection succeeds. If the TOE does mandate the presence of the SAN extension, this Test shall be omitted.

High-Level Test Description
Using the TOE TLS client, connect to the TLS server in the environment. The server will return an X.509 certificate which has a CN that matches and is missing the SAN. The TLS client is expected to succeed for DNS and IPv4 names.
Findings: PASS – The evaluator confirmed that certificates which contain valid CN but no SAN values are accepted by the TOE TLS client.

- **Test 4**: The evaluator shall present a server certificate that contains a CN that does not match the reference identifier but does contain an identifier in the SAN that matches. The evaluator shall verify that the connection succeeds.

High-Level Test Description
Using the TOE TLS client, connect to the TLS server in the environment. The server will return an X.509 certificate which has a CN that does not match and a SAN which matches. The TLS client is expected to succeed. Repeat for each SAN identifier type.
Findings: PASS – The evaluator confirmed that certificates which contain valid SAN values even though the CN is incorrect will be correctly validated by the TOE TLS client.

- **Test 5**: The evaluator shall perform the following wildcard tests with each supported type of reference identifier. The support for wildcards is intended to be optional. If wildcards are supported, the first, second, and third tests below shall be executed. If wildcards are not supported, then the fourth test below shall be executed.

    1. **Test 5.1**: [conditional]: If wildcards are supported, the evaluator shall present a server certificate containing a wildcard that is not in the left-most label of the presented identifier (e.g. foo.*.example.com) and verify that the connection fails.

High-Level Test Description
Using the TOE TLS client, connect to the TLS server in the environment. The server will return an X.509 certificate which has an invalid wildcard as described in the test case. The connection will fail. The test will be repeated for a similar wildcard in the SAN extension.
Findings: PASS – The evaluator confirmed that certificates that contain wildcards in which the asterisk in not in the left-most label are rejected.

2. **Test 5.2**: [conditional]: If wildcards are supported, the evaluator shall present a server certificate containing a wildcard in the left-most label but not preceding the public suffix (e.g. *.example.com). The evaluator shall configure the reference identifier with a single left-most label (e.g. foo.example.com) and verify that the connection succeeds. The evaluator shall configure the reference identifier without a left-most label as in the certificate (e.g. example.com) and verify that the connection fails. The evaluator shall configure the reference identifier with two left-most labels (e.g. bar.foo.example.come) and verify that the connection fails.

High-Level Test Description
Using the TOE TLS client, connect to the TLS server in the environment with a reference identifier that is consistent with the server certificate. The server will return an X.509 certificate which has a single valid wildcard in the left-most position. The connection will succeed. Using the TOE TLS client, connect to the TLS server in the environment with a reference identifier that has one more and one less identifier compared to the server certificate. The server will return an X.509 certificate which has a single valid wildcard in the left-most position. The connection will fail. The test will be repeated for a similar wildcard in the SAN extension.
Findings: PASS – The evaluator confirmed that the TOE TLS client, when presented with certificates that contain valid wildcards will match reference names when only one left-most label is provided. When the reference name has more than one left-most label, the certificate will be rejected by the TOE TLS client.

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

High-Level Test Description
Using the TOE TLS client, connect to the TLS server in the environment. The server will return an X.509 certificate which has an invalid wildcard as described in the test case. The connection will fail. The test will be repeated for a similar wildcard in the SAN extension.
Findings: PASS – The evaluator confirmed that the TOE TLS client will fail to accept certificates which include a wildcard specification where the asterisk is to the immediate left of the public suffix.

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

Note:	Wildcards are supported and therefore this test case is N/A.

- **Test 6**: [conditional] If URI or Service name reference identifiers are supported, the evaluator shall configure the DNS name and the service identifier. The evaluator shall present a server certificate containing the correct DNS name and service identifier in the URIName or SRVName fields of the SAN and verify that the connection succeeds. The evaluator shall

repeat this test with the wrong service identifier (but correct DNS name) and verify that the connection fails.

Findings:	The TOE does not claim the use of SRV or URI identifiers and therefore this test is N/A.

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

Findings:	The TOE does not claim certificate pinning and therefore this test is N/A.

7.1.2.7 FCS_TLSC_EXT.1.3 TSS

128 If the selection for authorizing override of invalid certificates is made, then the evaluator shall ensure that the TSS includes a description of how and when user or administrator authorization is obtained. The evaluator shall also ensure that the TSS describes any mechanism for storing such authorizations, such that future presentation of such otherwise-invalid certificates permits establishment of a trusted channel without user or administrator action.

Findings:	FCS_TLSC_EXT.1.3 in section 5.3.2 of the [ST] does not make the selection to permit override exceptions.

7.1.2.8 FCS_TLSC_EXT.1.3 Tests

129 The evaluator shall demonstrate that using an invalid certificate (unless excepted) results in the function failing as follows, unless excepted:

- Test 1a: The evaluator shall demonstrate that a server using a certificate with a valid certification path successfully connects.

Note:	This test was conducted in FIA_X509_EXT.1 test 64.

- Test 1b: The evaluator shall modify the certificate chain used by the server in test 1a to be invalid and demonstrate that a server using a certificate without a valid certification path to a trust store element of the TOE results in an authentication failure.

Note:	This test was conducted in FIA_X509_EXT.1 test 64.

- Test 1c [conditional]: If the TOE trust store can be managed, the evaluator shall modify the trust store element used in Test 1a to be untrusted and demonstrate that a connection attempt from the same server used in Test 1a results in an authentication failure.

Note:	This test was conducted in FIA_X509_EXT.1 test 64.

- **Test 2**: The evaluator shall demonstrate that a server using a certificate which has been revoked results in an authentication failure.

Note:	This test was conducted in FIA_X509_EXT.1 test 66.

- **Test 3**: The evaluator shall demonstrate that a server using a certificate which has passed its expiration date results in an authentication failure.

Note:	This test was conducted in FIA_X509_EXT.1 test 65.

- **Test 4**: The evaluator shall demonstrate that a server using a certificate which does not have a valid identifier results in an authentication failure.

Note:	Testing of invalid identifiers has been conducted in FCS_TLSC_EXT.1.2.

7.1.2.9 FCS_TLSC_EXT.3.1 TSS

130 The evaluator shall verify that TSS describes the signature_algorithm extension and whether the required behavior is performed by default or may be configured.

Findings:	Section 6.4.3 of the [ST] describes the signature_algorithm extension, identifying signature algorithms, key sizes, and hashes. The [ST] indicates the behaviour is performed by default.

7.1.2.10 FCS_TLSC_EXT.3.1 Guidance Documentation

131 If the TSS indicates that the signature_algorithm extension must be configured to meet the requirement, the evaluator shall verify that AGD guidance includes configuration of the signature_algorithm extension.

Findings:	The TSS indicates the signature_algorithm extension behaviour is performed by default.

7.1.2.11 FCS_TLSC_EXT.3.1 Tests

132 The evaluator shall also perform the following tests:

- **Test 1**: The evaluator shall configure the server to send a certificate in the TLS connection that is not supported according to the Client's HashAlgorithm enumeration within the signature_algorithms extension (for example, send a certificate with a SHA-1 signature). The evaluator shall verify that the product disconnects after receiving the server's Certificate handshake message.

High-Level Test Description
Using the TOE TLS client, show that when connecting to the TLS server in the environment which presents a certificate with an unsupported signature algorithm, that the TLS client will terminate.
Findings: PASS – The evaluator confirmed the TOE rejects TLS connections when the certificates use an unsupported hash algorithm.

- **Test 2**: [conditional] If the client supports a DHE or ECDHE cipher suite, the evaluator shall configure the server to send a Key Exchange handshake message including a signature not supported according to the client's HashAlgorithm enumeration (for example, the server signed the Key Exchange parameters using a SHA-1 signature). The evaluator shall verify that the product disconnects after receiving the server's Key Exchange handshake message.

High-Level Test Description
Using the TOE TLS client, show that when connecting to the TLS server in the environment which presents a certificate with an unsupported signature hash algorithm, that the TLS client will terminate.
Findings: PASS – The evaluator confirmed the TOE TLS library is capable of rejecting TLS handshakes when the Server Key Exchange uses a signature hash algorithm not supported/claimed by the TOE client.

7.1.2.12 FCS_TLSC_EXT.5.1 TSS

133 The evaluator shall verify that TSS describes the Supported Groups Extension.

Findings:	Section 6.4.3 of the [ST] describes the supported groups extension, identifying signature key agreement algorithms proposed.

7.1.2.13 FCS_TLSC_EXT.5.1 Tests

134 The evaluator shall also perform the following test:

- Test FCS_TLSC_EXT.5:1: The evaluator shall configure a server to perform key exchange using each of the TOE's supported curves and/or groups. The evaluator shall verify that the TOE successfully connects to the server.

High-Level Test Description
Using a TLS client, connect to the TLS server in the environment and show that each claimed ECDHE curve can successfully be negotiated.
Findings: PASS – The evaluator confirmed that each supported group is successfully negotiated.

7.1.3 FCS_SSH_EXT.1 SSH Protocol

This section has been modified by TD 0732.

7.1.3.1 FCS_SSH_EXT.1.1 TSS

135 The evaluator shall ensure that the selections indicated in the ST are consistent with selections in this and subsequent components. Otherwise, this SFR is evaluated by activities for other SFRs.

Findings:	The evaluator verified that the selections are consistent with the components: 1) RFC 4344 claimed to support AES-CTR encryption modes which is consistent with the claims made in FCS_SSH_EXT.1.4; 2) RFC 5647 claimed to support AES-GCM encryption modes which is consistent with the claims made in FCS_SSH_EXT.1.4; 3) RFC 5656 claimed to support ECDSA based public key algorithms which is consistent with the claims made in FCS_SSH_EXT.1.2, FCS_SSHS_EXT.1.1 and FCS_SSHC_EXT.1; 4) RFC 6668 claimed to support HMAC SHA2 message digest algorithms which is consistent with the use of SHA2 HMACs in FCS_SSH_EXT.1.5;

5) RFC 8268 claimed to support additional MODP Diffie-Hellman groups (DH group 16/SHA512 and DH group 18/SHA512) for SSH key exchange which is consistent with the claims made in FCS_SSH_EXT.1.6;
6) RFC 8308 claimed to support extension message negotiation to permit the use of RSA2 signatures for public key authentication consistent with the claims made in FCS_SSH_EXT.1.2 and FCS_SSHS_EXT.1 and FCS_SSHC_EXT.1.
7) RFC 8332 claimed to support specific signature modes for RSA public key authentication algorithms consistent with the claims made in FCS_SSH_EXT.1.2 and FCS_SSHS_EXT.1 and FCS_SSHC_EXT.1.

7.1.3.2 FCS_SSH_EXT.1.1 Guidance

136 There are no guidance evaluation activities for this component. This SFR is evaluated by activities for other SFRs.

7.1.3.3 FCS_SSH_EXT.1.1 Tests

137 There are no test evaluation activities for this component.

7.1.3.4 FCS_SSH_EXT.1.2 TSS

138 The evaluator shall check to ensure that the authentication methods listed in the TSS are identical to those listed in this SFR component; and, ensure if password-based authentication methods have been selected in the ST then these are also described; and, ensure that if keyboard-interactive is selected, it describes the multifactor authentication mechanisms provided by the TOE.

Findings: Section 6.2.11 of the [ST] specifies that the TOE supports rsa-sha2-256, rsa-sha2-512, ecdsa-sha2-nistp384, ecdsa-sha2-nistp521, and password based authentication methods. This conforms to the selections in FCS_SSH_EXT.1.2.

Note that the TOE claims use of "password" as the only passed-based authentication method which has been found in the [ST] section 6.2.11.

7.1.3.5 FCS_SSH_EXT.1.2 Guidance

139 The evaluator shall check the guidance documentation to ensure the configuration options, if any, for authentication mechanisms provided by the TOE are described.

Findings: [AGD] Section 6.3 and [RHEL] section 9.2.1 describe managing user passwords. [AGD] section 5.1.1.1 describes configuring SSH public key authentication.

7.1.3.6 FCS_SSH_EXT.1.2 Tests

- **Test 1**: [conditional] If the TOE is acting as SSH Server:

    a. The evaluator shall use a suitable SSH Client to connect to the TOE, enable debug messages in the SSH Client, and examine the debug messages to determine that only the configured authentication methods for the TOE were offered by the server.

High-Level Test Description
Using an SSH client, connect to the TOE SSH server and review the SSH messages to determine that the set of authentication mechanisms being advertised by the TOE are consistent with the claims.
Findings: PASS – The evaluator confirmed the authentication mechanisms advertised by the TOE server are consistent with the ST claims.

b. [conditional] If the SSH server supports X509 based Client authentication options:

   a. The evaluator shall initiate an SSH session from a client where the username is associated with the X509 certificate. The evaluator shall verify the session is successfully established.

   b. Next the evaluator shall use the same X509 certificate as above but include a username not associated with the certificate. The evaluator shall verify that the session does not establish.

   c. Finally, the evaluator shall use the correct username (from step a above) but use a different X509 certificate which is not associated with the username. The evaluator shall verify that the session does not establish.

Findings:	The TOE does not claim the use of X.509 based client authentication options. Therefore this test is not applicable.

- **Test 2**: [conditional] If the TOE is acting as SSH Client, the evaluator shall test for a successful configuration setting of each authentication method as follows:

  a. The evaluator shall initiate a SSH session using the authentication method configured and verify that the session is successfully established.

  b. Next, the evaluator shall use bad authentication data (e.g. incorrectly generated certificate or incorrect password) and ensure that the connection is rejected.

140 Steps a-b shall be repeated for each independently configurable authentication method supported by the server.

High-Level Test Description
Using the SSH client in the TOE, cycle through each of the claimed authentication mechanisms against a remote non-TOE SSH server. In the first instance, perform the action with a correct credential and show that the connection succeeds. In the second instance, perform the action with an incorrect credential and show that the connection fails.
Findings: PASS – The evaluator confirmed that for each claimed authentication mechanism, the TOE SSH client is capable of successfully connecting using the mechanism when good credentials are provided and fails to connect when bad credentials are provided.

- **Test 3**: [conditional] If the TOE is acting as SSH Client, the evaluator shall verify that the connection fails upon configuration mismatch as follows:

a. The evaluator shall configure the Client with an authentication method not supported by the Server.

b. The evaluator shall verify that the connection fails.

141 If the Client supports only one authentication method, the evaluator can test this failure of connection by configuring the Server with an authentication method not supported by the Client.

142 In order to facilitate this test, it is acceptable for the evaluator to configure an authentication method that is outside of the selections in the SFR.

High-Level Test Description
Establish an SSH connection using “password” as authentication method from the TOE to an SSH server in the environment which does not support passwords. Verify that this authentication method failed.
Findings: PASS – The evaluator confirmed that the connection fails when the TOE SSH client tries to use a server unsupported authentication mechanism.

7.1.3.7 FCS_SSH_EXT.1.3 TSS

143 The evaluator shall check that the TSS describes how “large packets” are detected and handled.

Findings:	The evaluator found that the TSS in section 6.2.11 of the [ST] specifies that large SSH packets are detected by checking the “packet_length” field. If this field size is greater than 262,144 bytes (256 KB), the packets are dropped.

7.1.3.8 FCS_SSH_EXT.1.3 Tests

- **Test 1**: The evaluator shall demonstrate that the TOE accepts the maximum allowed packet size.

High-Level Test Description
Using a custom SSH client, generate an SSH packet which is at the maximum TOE server accepted size. Using a custom SSH server, generate an SSH packet which is at the maximum TOE client accepted size.
Findings: PASS – The evaluator confirmed that sending a packet of the maximum size is accepted by the TOE SSH server and the TOE SSH client.

- **Test 2**: This test is performed to verify that the TOE drops packets that are larger than size specified in the component.

  a. The evaluator shall establish a successful SSH connection with the peer.

  b. Next the evaluator shall craft a packet that is slightly larger than the maximum size specified in this component and send it through the established SSH connection to the TOE. The packet should not be greater than the maximum packet size + 16 bytes. If the packet is larger, the evaluator shall justify the need to send a larger packet.

c. The evaluator shall verify that the packet was dropped by the TOE. The method of verification will vary by the TOE. Examples include reviewing the TOE audit log for a dropped packet audit or observing the TOE terminates the connection.

High-Level Test Description
Using a custom SSH client, generate an SSH packet which is 16 bytes larger than the maximum TOE SSH server accepted size. Review the audit log and verify that the packet was dropped by the TOE. Using a custom SSH server, generate an SSH packet which is 16 bytes larger than the maximum TOE SSH client accepted size. Review the debugging output and verify that the packet was dropped by the TOE.
Findings: PASS – The evaluator confirmed that both the TOE SSH server and TOE SSH client drop the packet larger than maximum size permitted.

7.1.3.9 FCS_SSH_EXT.1.4 TSS

144 The evaluator will check the description of the implementation of SSH in the TSS to ensure the encryption algorithms supported are specified. The evaluator will check the TSS to ensure that the encryption algorithms specified are identical to those listed for this component.

Findings:	Section 6.2.11 of the [ST] specifies that encryption algorithms aes256-ctr and aes256-gcm@openssh.com are supported. This list is identical to the selections made in FCS_SSH_EXT.1.4.

7.1.3.10 FCS_SSH_EXT.1.4 Guidance

145 The evaluator shall check the guidance documentation to ensure that it contains instructions to the administrator on how to ensure that only the allowed mechanisms are used in SSH connections with the TOE.

Findings:	[AGD] Section 5 indicates the kickstart script and SCAP remediate set the necessary SSH parameters.

7.1.3.11 FCS_SSH_EXT.1.4 Tests

146 The evaluator shall perform the following tests.

147 If the TOE can be both a client and a server, these tests must be performed for both roles.

- **Test 1**: The evaluator must ensure that only claimed algorithms and cryptographic primitives are used to establish an SSH connection. To verify this, the evaluator shall establish an SSH connection with a remote endpoint. The evaluator shall capture the traffic exchanged between the TOE and the remote endpoint during protocol negotiation (e.g. using a packet capture tool or information provided by the endpoint, respectively). The evaluator shall verify from the captured traffic that the TOE offers only the algorithms defined in the ST for the TOE for SSH connections. The evaluator shall perform one successful negotiation of an SSH connection and verify that the negotiated algorithms were included in the advertised set. If the evaluator detects that not all algorithms defined in the ST for SSH are advertised by the TOE or the

TOE advertises additional algorithms not defined in the ST for SSH, the test shall be regarded as failed.

The data collected from the connection above shall be used for verification of the advertised hashing and shared secret establishment algorithms in FCS_SSH_EXT.1.5 and FCS_SSH_EXT.1.6 respectively.

High-Level Test Description
Using an SSH client, successfully connect to the TOE SSH server using a claimed cipher combination and witness the exchange of algorithms and capabilities. Show that the set advertised by the TOE server matches the claims. Terminate the session and show that the session is terminated. Using an SSH server, successfully connect using a TOE SSH client using a claimed cipher combination and witness the exchange of algorithms and capabilities. Show that the set advertised by the TOE client matches the claims. Terminate the session and show that the session is terminated.
Findings: PASS – The evaluator confirmed that the set of advertised ciphers and capabilities are consistent with those claimed in the ST and that the TOE successfully terminates sessions.

- **Test 2**: For the connection established in Test 1, the evaluator shall terminate the connection and observe that the TOE terminates the connection.

Note:	This was performed in the previous test case.

- **Test 3**: The evaluator shall configure the remote endpoint to only allow a mechanism that is not included in the ST selection. The evaluator shall attempt to connect to the TOE and observe that the attempt fails.

High-Level Test Description
Connect to the TOE over SSH using the 3des-cbc cipher and show it fails to successfully negotiate.
Findings: PASS – The evaluator confirmed that the TOE SSH client and server are unable to establish sessions with remote endpoints configured with mismatching cipher mechanisms.

7.1.3.12 FCS_SSH_EXT.1.5 TSS

148 The evaluator will check the description of the implementation of SSH in the TSS to ensure the hashing algorithms supported are specified. The evaluator will check the TSS to ensure that the hashing algorithms specified are identical to those listed for this component.

Findings:	Section 6.2.11 of the [ST] specifies that the TOE supports hmac-sha2-256, hmac-sha2-512 and implicit (for GCM ciphers) data integrity algorithms. This conforms to the selections in FCS_SSH_EXT.1.5.

7.1.3.13 FCS_SSH_EXT.1.5 Guidance

149 The evaluator shall check the guidance documentation to ensure that it contains instructions to the administrator on how to ensure that only the allowed mechanisms are used in SSH connections with the TOE.

Findings:	[AGD] Section 5 indicates the kickstart script and SCAP remediate set the necessary SSH parameters.

7.1.3.14 FCS_SSH_EXT.1.5 Tests

- **Test 1:** The evaluator shall use the test data collected in FCS_SSH_EXT.1.4, Test 1 to verify that appropriate mechanisms are advertised.

High-Level Test Description
Review the data collected as part of FCS_SSH_EXT.1.4 test 1 and verify that the supported hashing functions are advertised.
Findings: PASS – The evaluator confirmed that the advertised set of MACs from the client and server match the claims in the ST.

- **Test 2:** The evaluator shall configure an SSH peer to allow only a hashing algorithm that is not included in the ST selection. The evaluator shall attempt to establish an SSH connection and observe that the connection is rejected.

High-Level Test Description
Using the hmac-sha1 integrity algorithms (and a supported ciphersuite permitting its use), show that the algorithm is not supported.
Findings: PASS – The evaluator confirmed that the TOE SSH client and server are unable to establish sessions with remote endpoints configured with mismatching HMAC mechanisms.

7.1.3.15 FCS_SSH_EXT.1.6 TSS

150 The evaluator will check the description of the implementation of SSH in the TSS to ensure the shared secret establishment algorithms supported are specified. The evaluator will check the TSS to ensure that the shared secret establishment algorithms specified are identical to those listed for this component.

Findings:	Section 6.2.11 of the [ST] specifies that the TOE supports the diffie-hellman-group16-sha512, diffie-hellman-group18-sha512, ecdh-sha2-nistp384, and ecdh-sha2-nistp521 key exchange algorithms. This list conforms to the selections in FCS_SSH_EXT.1.6.

7.1.3.16 FCS_SSH_EXT.1.6 Guidance

151 The evaluator shall check the guidance documentation to ensure that it contains instructions to the administrator on how to ensure that only the allowed mechanisms are used in SSH connections with the TOE.

Findings:	[AGD] Section 5 indicates the kickstart script and SCAP remediate set the necessary SSH parameters.

7.1.3.17 FCS_SSH_EXT.1.6 Tests

- **Test 1**: The evaluator shall use the test data collected in FCS_SSH_EXT.1.4, Test 1 to verify that appropriate mechanisms are advertised.

High-Level Test Description
Review the data collected as part of FCS_SSH_EXT.1.4 test 1 and verify that the supported key exchange functions are advertised.
Findings: PASS – The evaluator confirmed that the advertised set of key exchange algorithms from the client and server match the claims in the ST.

- **Test 2**: The evaluator shall configure an SSH peer to allow only a key exchange method that is not included in the ST selection. The evaluator shall attempt to establish an SSH connection and observe that the connection is rejected.

High-Level Test Description
Using the Diffie-hellman-group14-sha1 key exchange algorithm, show that the algorithm is not supported.
Findings: PASS – The evaluator confirmed that the TOE SSH client and server are unable to establish sessions with remote endpoints configured with mismatching key exchange mechanisms.

7.1.3.18 FCS_SSH_EXT.1.7 TSS

152 The evaluator will check the description of the implementation of SSH in the TSS to ensure the KDFs supported are specified. The evaluator will check the TSS to ensure that the KDFs specified are identical to those listed for this component.

Findings:	The [ST] section 6.2.11 describes that SSH KDFs defined in RFC 4253 (section 7.2) and RFC 5656 (section 4) are specified. These are consistent with the claims made in FCS_SSH_EXT.1.7.

7.1.3.19 FCS_SSH_EXT.1.8 TSS

153 The evaluator shall check the TSS to ensure that if the TOE enforces connection rekey or termination limits lower than the maximum values that these lower limits are identified.

Findings:	The [ST] in section 5.3.2 claims that the TSF will enforce connection rekey. This is described in the TSS in section 6.2.11 of the [ST]. The TSS states in section 6.2.11 of the [ST] that the TOE server configuration has a lower limit lower than the specified maximum. The TSS in the same section also indicates that the TOE SSH client will rekey based on either 1 hour or 1 GB of data and the TOE SSH server will rekey based on 1 hour and 1 GB of data.

154 In cases where hardware limitation will prevent reaching data transfer threshold in less than one hour, the evaluator shall check the TSS to ensure it contains:

a. An argument describing this hardware-based limitation and

b. Identification of the hardware components that form the basis of such argument.

For example, if specific Ethernet Controller or Wi-Fi radio chip is the root cause of such limitation, these subsystems shall be identified.

Findings:	The TOE does not have a hardware limitation preventing reaching this data transfer threshold.

7.1.3.20 FCS_SSH_EXT.1.8 Guidance

155 The evaluator shall check the guidance documentation to ensure that if the connection rekey or termination limits are configurable, it contains instructions to the administrator on how to configure the relevant connection rekey or termination limits for the TOE.

Findings:	[AGD] Section 5 indicates the kickstart script and SCAP remediate set the necessary SSH parameters.

7.1.3.21 FCS_SSH_EXT.1.8 Tests

156 The test harness needs to be configured so that its connection rekey or termination limits are greater than the limits supported by the TOE -- it is expected that the test harness should not be initiating the connection rekey or termination.

- **Test 1**: Establish an SSH connection. Wait until the identified connection rekey limit is met. Observed that a connection rekey or termination is initiated. This may require traffic to periodically be sent, or connection keep alive to be set, to ensure that the connection is not closed due to an idle timeout.

High-Level Test Description
Using a custom tool, SSH into the TOE server and wait for the prescribed time to elapse. Witness that the TOE initiates the rekey operation after the time has elapsed, no later than 1 hour. Using a custom server, use the SSH client on the TOE to connect and wait for the prescribed time to elapse. Witness that the TOE client initiates a rekey operation after the time has elapsed, no later than 1 hour.
Findings: PASS – The evaluator confirmed the TOE is capable of rekeying before 1 hour of elapsed time.

- **Test 2**: Establish an SSH connection. Transmit data from the TOE until the identified connection rekey or termination limit is met. Observe that a connection rekey or termination is initiated.

High-Level Test Description
Using a custom client, SSH into the TOE server and receive more than 1 GB of data from the TOE. Witness that the TOE initiates the rekey operation before 1 GB has been received. Using a custom server, use the SSH client on the TOE to transmit more than 1GB of data to the custom server. Witness that the TOE client initiates a rekey operation before 1 GB of data has been transmitted.
Findings: PASS – The evaluator confirmed the TOE is capable of rekeying before 1 GB of data has been transmitted from the TOE.

- **Test 3**: Establish an SSH connection. Send data to the TOE until the identified connection rekey limit or termination is met. Observe that a connection rekey or termination is initiated.

High-Level Test Description
Using a custom client, SSH into the TOE server and transmit more than 1 GB of data to the TOE. Witness that the TOE initiates the rekey operation before 1 GB has been received. Using a custom server, use the SSH client on the TOE to connect and receive more than 1GB of data fro the custom server. Witness that the TOE client initiates a rekey operation before 1 GB of data has been received.
Findings: PASS – The evaluator confirmed the TOE is capable of rekeying before 1 GB of data has been received by the TOE.

7.1.4 FCS_SSHS_EXT.1 SSH Protocol – Server

This section has been modified by TD 0682.

7.1.4.1 FCS_SSHS_EXT.1 TSS

157 No activities.

7.1.4.2 FCS_SSHS_EXT.1 Guidance

158 The evaluator shall check the guidance documentation to ensure that it contains instructions to the administrator on how to ensure that only the allowed mechanisms are used in SSH connections with the TOE.

Findings: [AGD] Section 5 indicates the kickstart script and SCAP remediate set the necessary SSH parameters.

7.1.4.3 FCS_SSHS_EXT.1 Tests

159 The evaluator shall perform the following tests:

160 Test 1: The evaluator shall use a suitable SSH Client to connect to the TOE and examine the list of server host key algorithms in the SSH_MSG_KEXINIT packet sent from the server to the client to determine that only the configured server authentication methods for the TOE were offered by the server.

High-Level Test Description
Connect to the TOE SSH server and show that the KEXINIT messages from the server to the client contain an advertised set of host key algorithms that can be negotiated. The set of algorithms will match the claims in the [ST].
Findings: PASS – The evaluator confirmed that the SSH server advertises support only for the configured/claimed host key algorithms described in the [ST].

161 Test 2: The evaluator shall test for a successful configuration setting of each server authentication method as follows. The evaluator shall initiate a SSH session using the authentication method configured and verify that the session is successfully established. Repeat this process for each independently configurable server authentication method supported by the server.

High-Level Test Description
Using an SSH client, configure the client to successfully negotiate an explicit host key algorithm and show that the session can be established successfully.
Findings: PASS – The evaluator confirmed that an SSH client can successfully establish sessions using each of the claimed/configured server host key algorithms.

162 Test 3: The evaluator shall configure the peer to only allow an authentication mechanism that is not included in the ST selection. The evaluator shall attempt to connect to the TOE and observe that the TOE sends a disconnect message.

High-Level Test Description
Using an SSH client, configure the client to attempt to negotiate an explicit host key algorithm that is unsupported by the TOE and show that the session fails to be established.
Findings: PASS – The evaluator confirmed that the TOE SSH server will disconnect the session when the SSH client and server are unable to establish a session when the host key algorithm cannot be verified by the client.

7.1.5 FCS_SSHC_EXT.1 SSH Protocol – Client

7.1.5.1 FCS_SSHC_EXT.1 TSS

163 No activities.

7.1.5.2 FCS_SSHC_EXT.1 Guidance

164 The evaluator shall check the guidance documentation to ensure that it contains instructions to the administrator on how to ensure that only the allowed mechanisms are used in SSH connections with the TOE.

Findings:	[AGD] Section 5 indicates the kickstart script and SCAP remediate set the necessary SSH parameters.

7.1.5.3 FCS_SSHC_EXT.1 Tests

165 The evaluator shall perform the following tests:

166 Test 1: [conditional] If using a local database by associating each host name with its corresponding public key, the evaluator shall configure the TOE with only a single host name and corresponding public key in the local database. The evaluator shall verify that the TOE can successfully connect to the host identified by the host name.

High-Level Test Description
Clear the local database. Add the hostname/public key association to the local database file. Using the TOE SSH client, connect to the non-TOE SSH server and show that the connection is permitted without warning.
Findings: PASS – The evaluator confirmed that the TOE SSH client is capable of connecting to a remote SSH server without warning when there is an entry in the local database file.

167 Test 2: [conditional] If using a local database by associating each host name with its corresponding public key, the evaluator shall configure the TOE with only a single

host name and non-corresponding public key in the local database. The evaluator shall verify that the TOE fails to connect to a host not identified by the host name.

High-Level Test Description
Clear the local database. Add a valid hostname but invalid public key association to the local database file. Using the TOE SSH client, connect to the non-TOE SSH server and show that the connection issues a warning and does not connect.
Findings: PASS – The evaluator confirmed that when an invalid key is associated with a known host, the TOE SSH client will generate a warning and fail to connect.

168 Test 3: [conditional] If using a local database by associating each host name with its corresponding public key, the evaluator shall try to connect to a host not configured in the local database. The evaluator shall verify that the TOE either fails to connect to a host identified by the host name or there is a prompt provided to store the public key in the local database.

High-Level Test Description
Clear the local database. Using the TOE SSH client, connect to the non-TOE SSH server and show that the connection issues a warning and prompts to add the key to the local database.
Findings: PASS – The evaluator confirmed that when the known host database does not have an entry for the system, the TOE SSH client will generate a warning and prompt to connect.

169 Test 4: [conditional] If using a list of trusted certification authorities, the evaluator shall configure the TOE with only a single trusted certification authority corresponding to the host. The evaluator shall verify that the TOE can successfully connect to the host identified by the host name.

Findings:	The TOE does not claim the use of X.509 certificates for SSH authentication such that a database of trusted certificate authorities is needed and therefore this test is N/A.

170 Test 5: [conditional] If using a list of trusted certification authorities, the evaluator shall configure the TOE with only a single trusted certification authority that does not correspond to the host. The evaluator shall verify that the TOE fails to the host identified by the host name.

Findings:	The TOE does not claim the use of X.509 certificates for SSH authentication such that a database of trusted certificate authorities is needed and therefore this test is N/A.

8 Evaluation Activities for Security Assurance Requirements

8.1 Class ADV: Development

171 The information about the OS is contained in the guidance documentation available to the end user as well as the TSS portion of the ST. The OS developer must concur with the description of the product that is contained in the TSS as it relates to the functional requirements. The evaluation activities contained in Section 5.1 Security Functional Requirements should provide the ST authors with sufficient information to determine the appropriate content for the TSS section.

8.1.1 ADV_FSP.1 Basic Functional Specification (ADV_FSP.1)

172 The functional specification describes the TSFIs. It is not necessary to have a formal or complete specification of these interfaces. Additionally, because OSes conforming to this PP will necessarily have interfaces to the operational environment that are not directly invokable by OS users, there is little point specifying that such interfaces be described in and of themselves since only indirect testing of such interfaces may be possible. For this PP, the activities for this family should focus on understanding the interfaces presented in the TSS in response to the functional requirements and the interfaces presented in the AGD documentation. No additional "functional specification" documentation is necessary to satisfy the evaluation activities specified. The interfaces that need to be evaluated are characterized through the information needed to perform the assurance activities listed, rather than as an independent, abstract list.

8.1.1.1 Evaluation Activity:

173 There are no specific evaluation 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 Security Functional Requirements, 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 evaluation 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.

Findings:	The evaluator was able to perform all evaluation activities described for AGD, ATE, and AVA with the evaluation evidence; therefore, an adequate functional specification was provided.

8.2 Class AGD: Guidance Documents

174 The guidance documents will be provided with the ST. Guidance must include a description of how the IT personnel verifies that the operational environment can fulfill its role for the security functionality. The documentation should be in an informal style and readable by the IT personnel. Guidance must be provided for every operational environment that the product supports as claimed in the ST. This guidance includes instructions to successfully install the TSF in that environment; and Instructions to manage the security of the TSF as a product and as a component of the larger operational environment. Guidance pertaining to particular security functionality is also provided; requirements on such guidance are contained in the Evaluation Activities specified with each requirement.

8.2.1 AGD_OPE.1 Operational User Guidance (AGD_OPE.1)

8.2.1.1 Evaluation Activity:

175 Some of the contents of the operational guidance are verified by the evaluation activities in Section 5.1 Security Functional Requirements and evaluation of the OS according to the [CEM]. The following additional information is also required. If cryptographic functions are provided by the OS, the operational guidance ill contain instructions for configuring the cryptographic engine associated with the evaluated configuration of the OS. It will provide a warning to the administrator that use of other cryptographic engines was not evaluated nor tested during the CC evaluation of the OS. The documentation must describe the process for verifying updates to the OS by verifying a digital signature – this may be done by the OS or the underlying platform. The evaluator will 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 OS (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 OS will likely contain security functionality that does not fall in the scope of evaluation under this PP. The operational guidance will make it clear to an administrator which security functionality is covered by the evaluation activities.

Findings:	[AGD] Section 5 indicates the kickstart script and SCAP remediate set the necessary cryptographic parameters. [AGD] Section 1.3.4 provides the warnings about cryptographic engines that were not evaluated. [AGD] Section 5.5 provides instructions for verifying the digital signatures on updates. [AGD] Section 3.2.2 describes how the dnf tool or automatic update system can be used to check for, obtain, and install updates. dnf provides error codes and prints messages indicating the success or failure of the update. [AGD] Section 10.2.10 shows the audit logs generated when verifying software.

8.2.2 AGD_PRE.1 Preparative Procedures (AGD_PRE.1)

8.2.2.1 Evaluation Activity:

176 As indicated in the introduction above, there are significant expectations with respect to the documentation—especially when configuring the operational environment to support OS functional requirements. The evaluator will check to ensure that the guidance provided for the OS adequately addresses all platforms claimed for the OS in the ST.

Findings:	[AGD] Sections 2, 3, and 4 provide instructions for configuring the operational environment to support initial installation of the TOE on each supported platform. [AGD] Sections 5 and 6 provide ongoing guidance for the administrator regarding configurations that must be performed and warnings about situations that may lead to an insecure state.

8.3 Class ALC: Life-cycle Support

177 At the assurance level provided for OSes conformant to this PP, life-cycle support is limited to end-user-visible aspects of the life-cycle, rather than an examination of the OS vendor'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 is a reflection on the information to be made available for evaluation at this assurance level.

8.3.1 ALC_CMC.1 Labelling of the TOE (ALC_CMC.1)

178 This component is targeted at identifying the OS such that it can be distinguished from other products or versions from the same vendor and can be easily specified when being procured by an end user.

8.3.1.1 Evaluation Activity:

179 The evaluator will 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.

Findings:	The [ST] in section 1.1 identifies the TOE as Red Hat Enterprise Linux 9.0 EUS. The version number of 9.0 with the “EUS” modifier is the version which meets the requirements.

180 Further, the evaluator will check the AGD guidance and OS 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 OS, the evaluator will examine the information on the web site to ensure that the information in the ST is sufficient to distinguish the product.

Findings:	The TOE itself, [AGD], and [ST] all consistently identify the TOE. Reviewing the Red Hat website shows the information in the ST is sufficient to distinguish the product.

8.3.2 ALC_CMS.1 TOE CM Coverage (ALC_CMS.1)

181 Given the scope of the OS and its associated evaluation evidence requirements, this component's evaluation activities are covered by the evaluation activities listed for ALC_CMC.1.

8.3.2.1 Evaluation Activity:

182 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 OS is specifically identified and that this identification is consistent in the ST and in the AGD guidance (as done in the evaluation 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.

Findings:	These evaluation activities are covered by the evaluation activities listed for ALC_CMC.1.

183 The evaluator will 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 will provide information on how to configure the environment to ensure that buffer overflow protection mechanisms in the environment(s) are invoked (e.g., compiler and linker flags). The evaluator will ensure that this documentation also includes an indication of whether such protections are on by default, or have to be specifically enabled. The evaluator will 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.

Findings:	[ST] section 6.6.3 identifies gcc compiler and the stack-protector-strong flag for configuring buffer overflow protections.

8.3.3 ALC_TSU_EXT.1 Timely Security Updates

184 This component requires the OS developer, in conjunction with any other necessary parties, to provide information as to how the end-user devices are updated to address security issues in a timely manner. The documentation describes the process of providing updates to the public from the time a security flaw is reported/discovered, to the time an update is released. This description includes the parties involved (e.g., the developer, carriers(s)) and the steps that are performed (e.g., developer testing, carrier testing), including worst case time periods, before an update is made available to the public.

8.3.3.1 Evaluation Activity:

185 The evaluator will 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 will verify that this description addresses the entire application. The evaluator will also verify that, in addition to the OS developer's process, any third-party processes are also addressed in the description. The evaluator will also verify that each mechanism for deployment of security updates is described.

Findings:	Section 6.10 of the [ST] provides a description of the developer's "timely security update methodology". This methodology describes a process used to create and deploy security updates. Red Hat engages with various partners, vendors, researchers, and community coordinators to disclose newly discovered vulnerabilities in a timely manner that takes into account the complexity and severity of each vulnerability and any collaborative efforts with stakeholders to produce coordinated and responsible disclosures and remediation guidance. The TOE only provides a single mechanism for deployment of updates.

186 The evaluator will 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 OS patching this vulnerability, to include any third-party or carrier delays in deployment. The evaluator will verify that this time is expressed in a number or range of days.

Findings:	[ST] Section 6.10.1 indicates information about security issues are communicated though the Red Hat CVE database and security advisories to active subscription holders. Advisories are provided through the rhsa-announce mailing list. Security updates are delivered via the standard update mechanism described in FPT_TUD_EXT.1.

187 The evaluator will verify that this description includes the publicly available mechanisms (including either an email address or website) for reporting security issues related to the OS. The evaluator will 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.

Findings:	[ST] Section 6.10.1 and [AGD] Section 9 include a public email address that can be used to communicate security issues and that the communications are encrypted using a GPG public key found at https://access.redhat.com/security/team/contact.

8.4 Class ATE: Tests

8.4.1 ATE_IND.1 Independent Testing - Conformance (ATE_IND.1)

188 Testing is performed to confirm the functionality described in the TSS as well as the administrative (including configuration and operational) documentation provided. The focus of the testing is to confirm that the requirements specified in Section 5.1 Security Functional Requirements being met, although some additional testing is specified for SARs in Section 5.2 Security Assurance Requirements. The evaluation activities identify the additional testing activities associated with these components. The evaluator produces a test report documenting the plan for and results of testing, as well as coverage arguments focused on the platform/OS combinations that are claiming conformance to this PP. Given the scope of the OS and its associated evaluation evidence requirements, this component's evaluation activities are covered by the evaluation activities listed for ALC_CMC.1.

8.4.1.1 Evaluation Activity:

189 The evaluator will prepare a test plan and report documenting the testing aspects of the system, including any application crashes during testing. The evaluator will 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 evaluation activities.

Findings:	The evaluator constructed and executed a test plan which was submitted as part of this evaluation. During the testing, no application crashes were reported. The evaluator ensured that all testing actions in the [CEM] and the body of the PP evaluation activities were represented within the test plan and suitably replicated within this public-facing document.

190 While it is not necessary to have one test case per test listed in an evaluation 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 OS and its platform.

191 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). 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.

Findings:	The evaluator's test plan provided full coverage of the applicable testing requirements. The test plan has a section devoted to the platforms under test and any equivalency arguments to justify platforms which were not explicitly tested. The test plan further provides information on the testing environment including setup necessary beyond the AGD documentation. For each test, a consideration of prerequisites for the test case was provided. For any prerequisites which deviated from the evaluated configuration (e.g., drivers, tools, debug modes, etc.) the test case includes a technical justification for why the specific setup will not adversely affect the results. The test plan describes the specific configuration of the cryptographic engine in use, if such functionality is provided. The test plan includes high level objectives – which are replicated within this public facing document – and more granular test steps needed to achieve that objective.

192 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 will 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.

Findings:	The test plan includes the expected results and separate test evidence documents provide the actual results.

8.5 Class AVA: Vulnerability Assessment

193 For the first generation of this protection profile, the evaluation lab is expected to survey open sources to discover what vulnerabilities have been discovered in these types of products. In most cases, these vulnerabilities will require sophistication beyond that of a basic attacker. Until penetration tools are created and uniformly distributed to the evaluation labs, the evaluator will not be expected to test for these vulnerabilities in the OS. The labs will be expected to comment on the likelihood of these vulnerabilities given the documentation provided by the vendor. This information will be used in the development of penetration testing tools and for the development of future protection profiles.

8.5.1 AVA_VAN.1 Vulnerability Survey (AVA_VAN.1)

8.5.1.1 Evaluation Activity (Documentation):

194 The evaluator will 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 documents the sources consulted and the vulnerabilities found in the report.

195 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.

Findings: The evaluator produced a separate report of the vulnerability assessment, which included a search of the public information. All sources were recorded in that separate report.

The vulnerability report included an assessment of each identified potential vulnerability and an assessment of whether the attack vector(s) was/were applicable.

A sanitized version of the vulnerability public survey results is replicated below.

This survey was last conducted on December 14, 2023.

The National Vulnerability Database (NVD) was searched for potential vulnerabilities using the TOE name and the following components of the TOE:

- auditd
- bzip
- chrony
- cURL
- dnf
- fapolicyd
- firewalld
- gpgme
- grub
- gnutls
- gzip
- Linux Kernel
- lz
- lzo
- OpenSSH
- OpenSSL
- PAM
- rpm
- sudo
- tar

The following vulnerabilities were identified, and the vendor plans to release patches for the vulnerabilities in a January 23, 2024 update to the TOE:

- CVE-2023-5717
- CVE-2023-5178
- CVE-2023-4622
- CVE-2023-38409
- CVE-2023-3567
- CVE-2023-3268
- CVE-2023-1195
- CVE-2022-36879
- CVE-2022-0480
- CVE-2023-2650

All other potential vulnerabilities were determined to be not applicable to the TOE.

The evaluation team determined, that based on these searches, no vulnerabilities (other than the above list of residual vulnerabilities) exist which are exploitable by attackers with Basic Attack Potential.