Red Hat Enterprise Linux 9.0 EUS

Security Target

Version 2.1

August 2024

Document prepared by

www.lightshipsec.com

Document History

Version	Date	Author	Description
2.0	23 FEB 2024	G. NICKEL	Release for Assurance Maintenance
2.1	22 AUG 2024	G. NICKEL	Update Guidance document version references

Table of Contents

List of Tables

1 Introduction

1.1 Overview

1 This Security Target (ST) defines the Red Hat Enterprise Linux 9.0 EUS Target of Evaluation (TOE) for the purposes of Common Criteria (CC) evaluation.

2 Red Hat Enterprise Linux 9.0 EUS is an open-source operating system that supports a general-purpose computing environment for multiple users and applications.

1.2 Identification

Table 1: Evaluation identifiers

Target of Evaluation	Red Hat Enterprise Linux 9.0 EUS Build: cc-config-9.0-2
Security Target	Red Hat Enterprise Linux 9.0 EUS Security Target, v2.1

1.3 Conformance Claims

3 This ST supports the following conformance claims:

a) CC version 3.1 revision 5

b) CC Part 2 extended

c) CC Part 3 extended

d) Protection Profile for General Purpose Operating Systems, Version 4.3 (PP_OS_V4.3)

e) Functional Package for Secure Shell (SSH), Version 1.0 (PKG_SSH_V1.0)

f) Functional Package for Transport Layer Security (TLS) 1.1 (PKG_TLS_V1.1)

g) NIAP Technical Decisions per Table 2

Table 2: NIAP 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

TD #	Name	Source	Applicability Rationale
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
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

TD #	Name	Source	Applicability Rationale
TD0812	Updated CC Conformance Claims in PP_OS_V4.3	PP_OS_V4.3	Applicable

1.4 Terminology

Table 3: Terminology

Term	Definition
ACL	Access Control List
AES	Advanced Encryption Standard
ASLR	Address Space Layout Randomization
CAVP	Cryptographic Algorithm Validation Program
CBC	Cipher Block Chaining
CC	Common Criteria
CLI	Command Line Interface
CMC	Certificate Management over CMS
CMS	Cryptographic Message Syntax
DAC	Discretionary Access Control
DNS	Domain Name System
DRBG	Deterministic Random Bit Generator
DSS	Digital Signature Standard
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
EKU	Extended Key Usage
EUS	Extended Update Support
FIPS	Federal Information Processing Standards
GCM	Galois Counter Mode
HMAC	Hash-based Message Authentication Code

Term	Definition
HTTPS	Hypertext Transfer Protocol Secure
IP	Internet Protocol
LAF	Lightweight Audit Framework
NIAP	Nation Information Assurance Partnership
NIST	National Institute of Standards and Technology
OID	Object Identifier
OS	Operating System
OSP	Organizational Security Policy
PP	Protection Profile
RA	Registration Authority
RFC	Request for Comments
RSA	Rivest, Shamir, & Adleman
SAR	Security Assurance Requirement
SFR	Security Functional Requirement
SHA	Secure Hash Algorithm
S/MIME	Secure/Multi-purpose Internet Mail Extensions
SSH	Secure Shell
ST	Security Target
TSF	TOE Security Function
TOE	Target of Evaluation
TLS	Transport Layer Security
TSS	TOE Summary Specification
UEFI	Unified Extensible Firmware Interface

2 TOE Description

2.1 Type

4 The TOE is an open-source, general purpose operating system (OS) that supports multiple users, user permissions, access controls, and cryptographic functionality.

2.2 Usage

5 The expected use cases (as defined by PP_OS_V4.3) for the TOE are:

a) **Server System.** The OS provides a platform for server-side services, either on physical or virtual hardware.

b) **Cloud System.** The OS provides a platform for providing cloud services running on physical or virtual hardware.

6 Users interact with the TOE locally (console) or remotely (SSH) via a CLI.

2.3 Security Functions / Logical Scope

7 The TOE provides the following security functions:

a) **Security Audit.** The TOE generates and stores security relevant audit events. These logs are stored locally and are protected by restricting access to system administrators only.

b) **Cryptographic Support.** The TOE implements cryptographic operations in support of its security functions. Relevant CAVP certificates are listed in Table 4.

c) **User Data Protection.** The TOE implements access controls to prevent unauthorized access to files and directories.

d) **Identification and Authentication.** The TOE supports password and public-key authentication. The TOE supports a configurable password and account lockout policy.

e) **Security Management.** The security management facilities provided by the TOE are usable by authorized users and/or authorized administrators to modify the configuration of TSF.

f) **TOE Access.** The TOE displays informative banners before users are allowed to establish a session.

g) **Protection of the TSF.** The TOE implements self-protection mechanisms that protect the security mechanisms of the TOE as well as software executed by the TOE. The following kernel-space isolation and TSF self-protection mechanisms are implemented and enforced (full details are provided in the TSS):

   i) Address Space Layout Randomization for user space code.

   ii) Kernel and user-space ring-based separation of processes

   iii) Stack buffer overflow protection using stack canaries.

   iv) Secure Boot ensures that the boot chain up to and including the kernel together with the boot image (initramfs) is not tampered with.

v) Updates to the operating system are only installed after their signatures have been successfully validated.

vi) Application Allow-lists restrict execution to known/trusted applications.

h) **Trusted Path/Channels.** The TOE supports TLSv1.2 and SSHv2 to secure remote communications. Both protocols may be used for communications with remote IT entities. Remote administration is only supported using SSHv2.

Table 4: CAVP Certificates

Module	Services	Operational Environment	CAVP
Linux Kernel Crypto API 5.14.0	Provides DRBG for OS applications and for seeding OpenSSL	Intel Xeon Silver 4216 (Cascade Lake)	A4770
		Z16	
		Power10	
OpenSSL 3.0.1	All other TOE cryptographic operations	Intel Xeon Silver 4216 (Cascade Lake)	A4771
		Z16	
		Power10	

2.4 TOE Scope

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

a) Red Hat Enterprise Linux 9.0 Extended Update Support, build cc-config-9.0-2

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

10 The physical boundary of the TOE as it pertains to the evaluated and tested configuration is described in Table 5.

2.4.1 Guidance Documents

11 The TOE includes the following guidance documents (PDF):

a) [ST] Red Hat Enterprise Linux 9.0 EUS Security Target, Version 2.1

b) [AGD] Red Hat Enterprise Linux 9.0 EUS Common Criteria Guide, Version 2.1

c) [RHEL] Red Hat Enterprise Linux 9, Configuring Basic System Settings, 2023-11-08

This document is available for download at:

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html/configuring_basic_system_settings/index

2.4.2 Non-TOE Components

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

a) **Update Server.** The TOE receives updates from an organization's local repository via TLS.

b) **SSH Server.** The TOE is capable of securely communicating with an SSHv2 server.

c) **SSH Client.** The TOE is capable of securely communicating with an SSHv2 client.

d) **Compute Platform.** The TOE requires a compute platform meeting the following specifications:

i) Intel Xeon Silver x86-64 UEFI platforms (of Cascade Lake microarchitecture)

ii) IBM z16 (LPAR) platforms

iii) Power10 PowerVM (LPAR) platforms

13 Platforms that meet the above specifications and were tested with the TOE in this evaluation are listed in Table 5.

Table 5: Tested Platforms

Vendor	Model	CPU
Dell	PowerEdge R440	Xeon Silver 4216 (Cascade Lake)
IBM	z16 3931-A01	IBM z16
IBM	POWER10 9080-HEX	Power10

* Testing performed on a logical partition (LPAR) of the z16 and Power10 processors.

2.4.3 Functions not included in the TOE Evaluation

14 The following product functions are not included within the scope of the evaluation:

a) SELinux Mandatory Access Control System

b) OS Virtualization Infrastructure

c) Containerization Infrastructure

d) Gnome desktop environment

3 Security Problem Description

15 The Security Problem Description has been reproduced from the PP_OS_V4.3 Protection Profile, PKG_TLS_V1.1 Functional Package, and PKG_SSH_V1.0 Functional Package for the convenience of the reader. The SPD describes the threats the TOE is expected to address, assumptions about the operational environment, and any organizational security policies (OSP's) that the TOE is expected to enforce.

3.1 Threats

Table 6: Threats (PP_OS_V4.3)

Identifier	Description
T.NETWORK_ATTACK	An attacker is positioned on a communications channel or elsewhere on the network infrastructure. Attackers may engage in communications with applications and services running on or part of the OS with the intent of compromise. Engagement may consist of altering existing legitimate communications.
T.NETWORK_ EAVESDROP	An attacker is positioned on a communications channel or elsewhere on the network infrastructure. Attackers may monitor and gain access to data exchanged between applications and services that are running on or part of the OS.
T.LOCAL_ATTACK	An attacker may compromise applications running on the OS. The compromised application may provide maliciously formatted input to the OS through a variety of channels including unprivileged system calls and messaging via the file system.
T.LIMITED_ PHYSICAL_ACCESS	An attacker may attempt to access data on the OS while having a limited amount of time with the physical device.

16 No additional threats have been identified in PKG_SSH_V1.0 and PKG_TLS_v1.1.

3.2 Assumptions

Table 7: Assumptions (PP_OS_V4.3)

Identifier	Description
A.PLATFORM	The OS relies upon a trustworthy computing platform for its execution. This underlying platform is out of scope of this PP.
A.PROPER_USER	The user of the OS is not willfully negligent or hostile,and uses the software in compliance with the applied enterprise security policy. At the same time, malicious software could act as the user, so requirements which confine malicious subjects are still in scope.
A.PROPER_ADMIN	The administrator of the OS is not careless, willfully negligent or hostile, and administers the OS within compliance of the applied enterprise security policy.

17 No additional assumptions have been identified in PKG_SSH_V1.0 and PKG_TLS_v1.1.

3.3 Organizational Security Policies

18 The PP and Functional Packages do not define any Organizational Security Policies (OSP's).

4 Security Objectives

19 The security objectives are reproduced from section 4 of PP_OS_V4.3.

20 No security objectives have been identified in PKG_SSH_V1.0 and PKG_TLS_v1.1.

4.1 Security Objectives for the TOE

Table 8: Security Objectives for the TOE (PP_OS_V4.3)

Identifier	Description
O.ACCOUNTABILITY	Conformant OSes ensure that information exists that allows administrators to discover unintentional issues with the configuration and operation of the operating system and discover its cause. Gathering event information and immediately transmitting it to another system can also enable incident response in the event of system compromise.
O.INTEGRITY	Conformant OSes ensure the integrity of their update packages. OSes are seldom if ever shipped without errors, and the ability to deploy patches and updates with integrity is critical to enterprise network security. Conformant OSes provide execution environment-based mitigations that increase the cost to attackers by adding complexity to the task of compromising systems.
O.MANAGEMENT	To facilitate management by users and the enterprise, conformant OSes provide consistent and supported interfaces for their security-relevant configuration and maintenance. This includes the deployment of applications and application updates through the use of platform-supported deployment mechanisms and formats, as well as providing mechanisms for configuration and application execution control.
O.PROTECTED_ STORAGE	To address the issue of loss of confidentiality of credentials in the event of loss of physical control of the storage medium, conformant OSes provide data-at-rest protection for credentials. Conformant OSes also provide access controls which allow users to keep their files private from other users of the same system.
O.PROTECTED_ COMMS	To address both passive (eavesdropping) and active (packet modification) network attack threats, conformant OSes provide mechanisms to create trusted channels for CSP and sensitive data. Both CSP and sensitive data should not be exposed outside of the platform.

4.2 Security Objectives for the Operational Environment

21 Security objectives for the Operational Environment assist the TOE in correctly providing its security functionality. These objectives track with the assumptions about the TOE operational environment.

Table 9: Security Objectives for the Operational Environment (PP_OS_V4.3)

Identifier	Description
OE.PLATFORM	The OS relies on being installed on trusted hardware.
OE.PROPER_USER	The user of the OS is not willfully negligent or hostile, and uses the software within compliance of the applied enterprise security policy. Standard user accounts are provisioned in accordance with the least privilege model. Users requiring higher levels of access should have a separate account dedicated for that use.
OE.PROPER_ADMIN	The administrator of the OS is not careless, willfully negligent or hostile, and administers the OS within compliance of the applied enterprise security policy.

4.3 Security Objectives Rationale

22 This section describes how the assumptions, threats, and organizational security policies map to the security objectives. The below table is reproduced from PP_OS_4.3 Section 4.3.

Table 10: Security Objectives Rationale

Threat, Assumption, or OSP	Security Objectives	Rationale
T.NETWORK_ ATTACK	O.PROTECTED_ COMMS	The threat T.NETWORK_ATTACK is countered by O.PROTECTED_COMMS as this provides for integrity of transmitted data.
	O.INTEGRITY	The threat T.NETWORK_ATTACK is countered by O.INTEGRITY as this provides for integrity of software that is installed onto the system from the network.
	O.MANAGEMENT	The threat T.NETWORK_ATTACK is countered by O.MANAGEMENT as this provides for the ability to configure the OS to defend against network attack.
	O.ACCOUNTABILITY	The threat T.NETWORK_ATTACK is countered by O.ACCOUNTABILITY as this provides a mechanism for the OS to report behavior that may indicate a network attack has occurred.

Threat, Assumption, or OSP	Security Objectives	Rationale
T.NETWORK_ EAVESDROP	O.PROTECTED_ COMMS	The threat T.NETWORK_EAVESDROP is countered by O.PROTECTED_COMMS as this provides for confidentiality of transmitted data.
	O.MANAGEMENT	The threat T.NETWORK_EAVESDROP is countered by O.MANAGEMENT as this provides for the ability to configure the OS to protect the confidentiality of its transmitted data.
T.LOCAL_ ATTACK	O.INTEGRITY	The objective O.INTEGRITY protects against the use of mechanisms that weaken the TOE with regard to attack by other software on the platform.
	O.ACCOUNTABILITY	The objective O.ACCOUNTABILITY protects against local attacks by providing a mechanism to report behavior that may indicate a local attack is occurring or has occurred.
T.LIMITED_ PHYSICAL_ ACCESS	O.PROTECTED_ STORAGE	The objective O.PROTECTED_STORAGE protects against unauthorized attempts to access physical storage used by the TOE.
A.PLATFORM	OE.PLATFORM	The operational environment objective OE.PLATFORM is realized through A.PLATFORM.
A.PROPER_ USER	OE.PROPER_USER	The operational environment objective OE.PROPER_USER is realized through A.PROPER_USER.
A.PROPER_ ADMIN	OE.PROPER_ADMIN	The operational environment objective OE.PROPER_ADMIN is realized through A.PROPER_ADMIN.

5 Security Requirements

23 This section identifies the Security Functional Requirements (SFRs) for the TOE. The SFRs included in this section are reproduced from PP_OS_V4.3, PKG_SSH_V1.0, and PKG_TLS_v1.1 with applicable selection and assignment operations completed.

5.1 Conventions

24 This document uses the following font conventions to identify the operations defined by the CC:

a) **Assignment.** Indicated with italicized text.

b) **Refinement.** Indicated with bold text and strikethroughs.

c) **Selection.** Indicated with underlined text.

d) **Assignment within a Selection:** Indicated with italicized and underlined text.

e) **Iteration.** Indicated by adding a string starting with "/" (e.g. "FCS_COP.1/Hash").

25 **Note:** Operations performed within the Security Target are denoted within brackets []. Operations shown without brackets are reproduced from the PP and Functional Packages.

5.2 Extended Components Definition

26 Refer to the Extended Components Definitions sections of the PP and Functional Packages as follows:

a) PP_OS_v4.3 – Appendix 'C'

b) PKG_SSH_v1.0 – No extended components definition listed.

c) PKG_TLS_v1.1 – No extended components definition listed.

5.3 Functional Requirements

Table 11: Summary of SFRs

Requirement	Title
FAU_GEN.1	Audit Data Generation (Refined)
FCS_CKM.1	Cryptographic Key Generation (Refined)
FCS_CKM.2	Cryptographic Key Establishment (Refined)
FCS_CKM_EXT.4	Cryptographic Key Destruction
FCS_COP.1/ENCRYPT	Cryptographic Operation - Encryption/Decryption (Refined)
FCS_COP.1/HASH	Cryptographic Operation - Hashing (Refined)
FCS_COP.1/SIGN	Cryptographic Operation - Signing (Refined)

Requirement	Title
FCS_COP.1/KEYHMAC	Cryptographic Operation - Keyed-Hash Message Authentication (Refined)
FCS_RBG_EXT.1/KCAPI	Random Bit Generation (Kernel)
FCS_RBG_EXT.1/OSSL	Random Bit Generation (OpenSSL)
FCS_STO_EXT.1	Storage of Sensitive Data
FCS_SSH_EXT.1	SSH Protocol
FCS_SSHC_EXT.1	SSH Protocol - Client
FCS_SSHS_EXT.1	SSH Protocol - Server
FCS_TLS_EXT.1	TLS Protocol
FCS_TLSC_EXT.1	TLS Client Protocol
FCS_TLSC_EXT.3	TLS Client Support for Signature Algorithms Extension
FCS_TLSC_EXT.5	TLS Client Support for Supported Groups Extension
FDP_ACF_EXT.1	Access Controls for Protecting User Data
FIA_AFL.1	Authentication failure handling (Refined)
FIA_UAU.5	Multiple Authentication Mechanisms (Refined)
FIA_X509_EXT.1	X.509 Certificate Validation
FIA_X509_EXT.2	X.509 Certificate Authentication
FMT_MOF_EXT.1	Management of security functions behavior
FMT_SMF_EXT.1	Specification of Management Functions
FPT_ACF_EXT.1	Access controls
FPT_ASLR_EXT.1/Xeon	Address Space Layout Randomization (Xeon)
FPT_ASLR_EXT.1/z16	Address Space Layout Randomization (z16)
FPT_ASLR_EXT.1/Power10	Address Space Layout Randomization (Power10)
FPT_SBOP_EXT.1	Stack Buffer Overflow Protection
FPT_SRP_EXT.1	Software Restriction Policies
FPT_TST_EXT.1	Boot Integrity

Requirement	Title
FPT_TUD_EXT.1	Trusted Update
FPT_TUD_EXT.2	Trusted Update for Application Software
FTA_TAB.1	Default TOE access banners
FTP_ITC_EXT.1	Trusted channel communication
FTP_TRP.1	Trusted Path

5.3.1 Security Audit (FAU)

FAU_GEN.1 Audit Data Generation (Refined)

FAU_GEN.1.1 The **OS** shall be able to generate an audit record of the following auditable events:

a) Start-up and shut-down of the audit functions;

b) All auditable events for the [*not specified*] level of audit; and [

c)

- *Authentication events (Success/Failure);*
- *Use of privileged/special rights events (Successful and unsuccessful security, audit, and configuration changes);*
- *Privilege or role escalation events (Success/Failure);*
- [
  - File and object events (Successful and unsuccessful attempts to create, access, delete, modify, modify permissions)
  - User and Group management events (Successful and unsuccessful add, delete, modify, disable, enable, and credential change)
  - Audit and log data access events (Success/Failure)
  - Cryptographic verification of software (Success/Failure)
  - Attempted application invocation with arguments (Success/Failure e.g. due to software restriction policy)
  - System reboot, restart, and shutdown events (Success/Failure)
  - Kernel module loading and unloading events (Success/Failure)
  - Administrator or root-level access events (Success/Failure)

- [specifically defined auditable events listed in Table 12].

]

].

FAU_GEN.1.2 The **OS** shall record within each audit record at least the following information:

a) Date and time of the event, type of event, subject identity (if applicable), and outcome (success or failure) of the event; and

b) For each audit event type, based on the auditable event definitions of the functional components included in the PP/ST, [*information specified in column 3 of Table 12*]

Table 12: SSH Auditable Events

Requirement	Auditable Events	Additional Audit Record Contents
FCS_SSH_EXT.1	[Failure to establish SSH connection]	[Reason for failure and Non-TOE endpoint of attempted connection (IP Address)]
FCS_SSH_EXT.1	[Establishment of SSH connection]	[Non-TOE endpoint of connection (IP Address)]
FCS_SSH_EXT.1	[Termination of SSH connection session]	[Non-TOE endpoint of connection (IP Address)]
FCS_SSH_EXT.1	[Dropping of packet(s) outside defined size limits]	[Packet size]
FCS_SSHC_EXT.1	No events specified.	N/A
FCS_SSHS_EXT.1	No events specified.	N/A

Application Note: This table has been modified by TD0777.

5.3.2 Cryptographic Support (FCS)

FCS_CKM.1 Cryptographic Key Generation (Refined)

FCS_CKM.1.1 The **OS** shall generate **asymmetric** cryptographic keys in accordance with a specified cryptographic key generation algorithm [

- 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
- 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

- FFC schemes using [safe primes that meet the following: NIST Special Publication 800-56A Revision 3, “Recommendation for Pair-Wise Key Establishment Schemes"]

].

FCS_CKM.2 Cryptographic Key Establishment (Refined)

FCS_CKM.2.1 The OS shall **implement functionality to perform cryptographic key establishment** in accordance with a specified cryptographic key **establishment** method: [

- Elliptic curve-based key establishment schemes that meets the following: NIST Special Publication 800-56A Revision 3, “Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography”
- 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”

].

FCS_CKM_EXT.4 Cryptographic Key Destruction

FCS_CKM_EXT.4.1 The OS shall destroy cryptographic keys and key material in accordance with a specified cryptographic key destruction method [

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

  ]
- For non-volatile memory that consists of [
  - the invocation of an interface provided by the underlying platform that [
    - logically addresses the storage location of the key and performs a [*[administrator specified number (default of 3) of]*] overwrite consisting of [pseudo-random pattern]

      ]

  ]

].

FCS_CKM_EXT.4.2 The OS shall destroy all keys and key material when no longer needed.

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

FCS_COP.1.1/ENCRYPT The **OS** shall perform [*encryption/decryption services for data*] in accordance with a specified cryptographic algorithm [

- **AES-CBC (as defined in NIST SP 800-38A)**
- **AES-CTR (as defined in NIST SP 800-38A)**

] and [

- **AES-GCM (as defined in NIST SP 800-38D)**

] and cryptographic key sizes 256-bit and [no other bit size] ~~that meet the following: [assignment: list of standards].~~

Application Note: This SFR has been modified by TD0712.

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

FCS_COP.1.1/HASH The **OS** shall perform [*cryptographic hashing services*] in accordance with a specified cryptographic algorithm [

- **SHA-256**
- **SHA-384**
- **SHA-512**

] and message digest sizes [

- **256 bits**
- **384 bits**
- **512 bits**

] that meet the following: [*FIPS Pub 180-4*].

Application Note: This SFR has been modified by TD0696.

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

FCS_COP.1.1/SIGN The **OS** shall perform [*cryptographic signature services (generation and verification)*] in accordance with a specified cryptographic algorithm **[**

- **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**
- **ECDSA schemes using "NIST curves" P-384 and [P-521] that meet the following: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Section 5**

] ~~and cryptographic key sizes [assignment: cryptographic algorithm] that meet the following. [assignment: list of standards].~~

Application Note: This SFR has been modified by TD0809.

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

FCS_COP.1.1/KEYHMAC The **OS** shall perform [*keyed-hash message authentication services*] in accordance with a specified cryptographic algorithm **[SHA-256, SHA-384, SHA-512] with key sizes [*256-bits, 384-bits, 512-bits*] and message digest** sizes [256 bits, 384 bits,

512 bits] that meet the following: [*FIPS Pub 198-1 The Keyed-Hash Message Authentication Code and FIPS Pub 180-4 Secure Hash Standard*].

FCS_RBG_EXT.1/KCAPI Random Bit Generation (Kernel)

FCS_RBG_EXT.1.1/KCAPI The OS shall perform all deterministic random bit generation (DRBG) services in accordance with NIST Special Publication 800-90A using [

- HMAC_DRBG (any)

] .

FCS_RBG_EXT.1.2/KCAPI The deterministic RBG used by the OS shall be seeded by an entropy source that accumulates entropy from a [

- software-based noise source

] with a minimum of 256 bits of entropy at least equal to the greatest security strength (according to NIST SP 800-57) of the keys and hashes that it will generate.

FCS_RBG_EXT.1/OSSL Random Bit Generation (OpenSSL)

FCS_RBG_EXT.1.1/OSSL The OS shall perform all deterministic random bit generation (DRBG) services in accordance with NIST Special Publication 800-90A using [

- CTR_DRBG (AES)

] .

FCS_RBG_EXT.1.2/OSSL The deterministic RBG used by the OS shall be seeded by an entropy source that accumulates entropy from a [

- software-based noise source

] with a minimum of 256 bits of entropy at least equal to the greatest security strength (according to NIST SP 800-57) of the keys and hashes that it will generate.

FCS_STO_EXT.1 Storage of Sensitive Data

FCS_STO_EXT.1.1 The OS shall implement functionality to encrypt sensitive data stored in non-volatile storage and provide interfaces to applications to invoke this functionality.

FCS_SSH_EXT.1 SSH Protocol

FCS_SSH_EXT.1.1 The TOE shall implement SSH acting as a [client, server] in accordance with that complies with RFCs 4251, 4252, 4253, 4254, [4344, 5647, 5656, 6668, 8268, 8308, 8332] and [*no other standard*].

FCS_SSH_EXT.1.2 The TSF shall ensure that the SSH protocol implementation supports the following authentication methods: [

- “password” (RFC 4252),

- “publickey” (RFC 4252): [
  - rsa-sha2-256 (RFC 8332),
  - rsa-sha2-512 (RFC 8332),
  - ecdsa-sha2-nistp384 (RFC 5656),
  - ecdsa-sha2-nistp521 (RFC 5656),

  ]

] and no other methods.

FCS_SSH_EXT.1.3 The TSF shall ensure that, as described in RFC 4253, packets greater than [*262,144 bytes*] in an SSH transport connection are dropped.

FCS_SSH_EXT.1.4 The TSF shall protect data in transit from unauthorised disclosure using the following mechanisms: [

- aes256-ctr (RFC 4344),
- aes256-gcm@openssh.com (RFC 5647)

] and no other mechanisms.

FCS_SSH_EXT.1.5 The TSF shall protect data in transit from modification, deletion, and insertion using: [

- hmac-sha2-256 (RFC 6668),
- hmac-sha2-512 (RFC 6668),
- Implicit

] and no other mechanisms.

FCS_SSH_EXT.1.6 The TSF shall establish a shared secret with its peer using: [

- diffie-hellman-group16-sha512 (RFC 8268),
- diffie-hellman-group18-sha512 (RFC 8268),
- ecdh-sha2-nistp384 (RFC 5656),
- ecdh-sha2-nistp521 (RFC 5656),

] and no other mechanisms.

FCS_SSH_EXT.1.7 The TSF shall use *SSH KDF* as defined in [

- RFC 4253 (Section 7.2),
- RFC 5656 (Section 4)

] to derive the following cryptographic keys from a shared secret: *session keys.*

FCS_SSH_EXT.1.8 The TSF shall ensure that [

- a rekey of the session keys

] occurs when any of the following thresholds are met:

- one hour connection time
- no more than one gigabyte of transmitted data, or
- no more than one gigabyte of received data.

FCS_SSHC_EXT.1 SSH Protocol - Client

FCS_SSHC_EXT.1.1 The TSF shall authenticate its peer (SSH server) using: [

- using a local database by associating each host name with a public key corresponding to the following list: [
  - rsa-sha2-256 (RFC 8332),
  - rsa-sha2-512 (RFC 8332),
  - ecdsa-sha2-nistp384 (RFC 5656),
  - ecdsa-sha2-nistp521 (RFC 5656),

  ],

] as described in RFC 4251 Section 4.1.

FCS_SSHS_EXT.1 SSH Protocol - Server

FCS_SSHS_EXT.1.1 The TSF shall authenticate itself to its peer (SSH Client) using: [

- rsa-sha2-256 (RFC 8332),
- rsa-sha2-512 (RFC 8332),
- ecdsa-sha2-nistp384 (RFC 5656),
- ecdsa-sha2-nistp521 (RFC 5656),

].

FCS_TLS_EXT.1 TLS Protocol

FCS_TLS_EXT.1.1 The product shall implement [

- TLS as a client,

].

FCS_TLSC_EXT.1 TLS Client Protocol

FCS_TLSC_EXT.1.1 The product shall implement TLS 1.2 (RFC 5246) and [no earlier TLS versions] as a client that supports the cipher suites [

- TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5288,
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289,
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289

] and also supports functionality for [

- none

].

FCS_TLSC_EXT.1.2 The product shall verify that the presented identifier matches the reference identifier according to RFC 6125.

FCS_TLSC_EXT.1.3 The product shall not establish a trusted channel if the server certificate is invalid [

- with no exceptions,

].

Application Note: This SFR has been modified by TD0442.

FCS_TLSC_EXT.3 TLS Client Support for Signature Algorithms Extension

FCS_TLSC_EXT.3.1 The product shall present the signature_algorithms extension in the Client Hello with the supported_signature_algorithms value containing the following hash algorithms: [SHA256, SHA384, SHA512] and no other hash algorithms.

FCS_TLSC_EXT.5 TLS Client Support for Supported Groups Extension

FCS_TLSC_EXT.5.1 The product shall present the Supported Groups Extension in the Client Hello with the supported groups [

- secp384r1
- secp521r1

].

5.3.3 User Data Protection (FDP)

FDP_ACF_EXT.1 Access Controls for Protecting User Data

FDP_ACF_EXT.1.1 The OS shall implement access controls which can prohibit unprivileged users from accessing files and directories owned by other users.

5.3.4 Identification and Authentication (FIA)

FIA_AFL.1 Authentication Failure Handling (Refined)

FIA_AFL.1.1 The **OS** shall detect when [

- an administrator configurable positive integer within [*0 (disabled) – 65,535*]

] unsuccessful authentication attempts occur related to **events with** [

- authentication based on user name and password

] .

FIA_AFL.1.2 When the defined number of unsuccessful authentication attempts for an account has been **met**, the **OS** shall: **[Account Lockout].**

FIA_UAU.5 Multiple Authentication Mechanisms (Refined)

FIA_UAU.5.1 The **OS** shall provide the following authentication mechanisms [

- **authentication based on username and password**

- **for use in SSH only, SSH public key-based authentication as specified by the Functional Package for Secure Shell (SSH), version 1.0**

] to support user authentication.

FIA_UAU.5.2 The **OS** shall authenticate any user's claimed identity according to the [*username and password: used at the local console and over SSH: the TOE locally verifies the password hash matches the stored password hash associated with the provided username;*

SSH public key: used over SSH: the TOE verifies the signature can be verified using a public key in the authorized_keys file associated with the provided username].

FIA_X509_EXT.1 X.509 Certificate Validation

FIA_X509_EXT.1.1 The OS shall implement functionality to validate certificates in accordance with the following rules:

- RFC 5280 certificate validation and certificate path validation
- The certificate path must terminate with a trusted CA certificate
- The OS shall validate a certificate path by ensuring the presence of the basicConstraints extension, that the CA flag is set to TRUE for all CA certificates, and that any path constraints are met.
- The TSF shall validate that any CA certificate includes "Certificate Signing" as a purpose the key usage field
- The OS shall validate the revocation status of the certificate using [an OCSP TLS Status Request Extension (OCSP stapling) as specified in RFC 6066] with [no exceptions]
- The OS shall validate the extendedKeyUsage field according to the following rules:
    - Certificates used for trusted updates and executable code integrity verification shall have the Code Signing Purpose (id-kp 3 with OID 1.3.6.1.5.5.7.3.3) in the extendedKeyUsage field.
    - Server certificates presented for TLS shall have the Server Authentication purpose (id-kp 1 with OID 1.3.6.1.5.5.7.3.1) in the extendedKeyUsage field.
    - Client certificates presented for TLS shall have the Client Authentication purpose (id-kp 2 with OID 1.3.6.1.5.5.7.3.2) in the EKU field.
    - S/MIME certificates presented for email encryption and signature shall have the Email Protection purpose (id-kp 4 with OID 1.3.6.1.5.5.7.3.4) in the EKU field.
    - OCSP certificates presented for OCSP responses shall have the OCSP Signing Purpose (id-kp 9 with OID 1.3.6.1.5.5.7.3.9) in the EKU field.
    - Server certificates presented for EST shall have the CMC Registration Authority (RA) purpose (id-kp-cmcRA

with OID 1.3.6.1.5.5.7.3.28) in the EKU field. (conditional)

FIA_X509_EXT.1.2 The OS shall only treat a certificate as a CA certificate if the *basicConstraints* extension is present and the CA flag is set to TRUE.

FIA_X509_EXT.2 X.509 Certificate Authentication

FIA_X509_EXT.2.1 The OS shall use X.509v3 certificates as defined by RFC 5280 to support authentication for [TLS, HTTPS] connections.

Application Note: This SFR has been modified by TD0789.

5.3.5 Security Management (FMT)

FMT_MOF_EXT.1 Management of Security Functions Behavior

FMT_MOF_EXT.1.1 The OS shall restrict the ability to perform the function indicated in the "Administrator" column in FMT_SMF_EXT.1.1 to the administrator.

FMT_SMF_EXT.1 Specification of Management Functions

FMT_SMF_EXT.1.1 The OS shall be capable of performing the following management functions:

Table 13: Management Functions

#	Management Function	Administrator	User
1	Enable/disable [session timeout]	X	-
2	Configure [session] inactivity timeout	X	-
3	Import keys/secrets into the secure key storage	X	
4	Configure local audit storage capacity	X	-
5	Configure minimum password length	X	-
6	Configure minimum number of special characters in password	X	-
7	Configure minimum number of numeric characters in password	X	-
8	Configure minimum number of uppercase characters in password	X	-
9	Configure minimum number of lowercase characters in password	X	-
10	Configure lockout policy for unsuccessful authentication attempts through [timeouts between attempts]	X	-

#	Management Function	Administrator	User
11	Configure host-based firewall	X	-
12	Configure name/address of directory server with which to bind	-	-
13	Configure name/address of remote management server from which to receive management settings	-	-
14	Configure name/address of audit/logging server to which to send audit/logging records	-	-
15	Configure audit rules	X	-
16	Configure name/address of network time server	X	-
17	Enable/disable automatic software update	X	-
18	Configure WiFi interface	-	-
19	Enable/disable Bluetooth interface	-	-
20	Enable/disable [*no other external interfaces*]	-	-
21	[*no other management functions*]	-	-

5.3.6 Protection of the TSF (FPT)

FPT_ACF_EXT.1 Access Controls

FPT_ACF_EXT.1.1 The OS shall implement access controls which prohibit unprivileged users from modifying:

- Kernel and its drivers/modules
- Security audit logs
- Shared libraries
- System executables
- System configuration files
- [*no other objects*]

FPT_ACF_EXT.1.2 The OS shall implement access controls which prohibit unprivileged users from reading:

- Security audit logs
- System-wide credential repositories
- [*no other objects*]

FPT_ASLR_EXT.1/Xeon Address Space Layout Randomization on Xeon Silver

FPT_ASLR_EXT.1.1/Xeon The OS shall always randomize process address space memory locations with [*[at least 29]*] bits of entropy except for [*no exceptions*].

FPT_ASLR_EXT.1/z16 Address Space Layout Randomization on IBM z16

FPT_ASLR_EXT.1.1/z16 The OS shall always randomize process address space memory locations with [*[at least 11]*] bits of entropy except for [*no exceptions*].

FPT_ASLR_EXT.1/Power10 Address Space Layout Randomization on Power10

FPT_ASLR_EXT.1.1/Power10 The OS shall always randomize process address space memory locations with [*[at least 14]*] bits of entropy except for [*no exceptions*].

FPT_SBOP_EXT.1 Stack Buffer Overflow Protection

FPT_SBOP_EXT.1.1 The OS shall [employ stack-based buffer overflow protections].

FPT_SRP_EXT.1 Software Restriction Policies

FPT_SRP_EXT.1.1 The OS shall restrict execution to only programs which match an administrator-specified [

- file path
- hash

] .

FPT_TST_EXT.1 Boot Integrity

FPT_TST_EXT.1.1 The OS shall verify the integrity of the bootchain up through the OS kernel and [

- no other executable code

] prior to its execution through the use of [

- a digital signature using a hardware-protected asymmetric key

] .

FPT_TUD_EXT.1 Trusted Update

FPT_TUD_EXT.1.1 The OS shall provide the ability to check for updates to the OS software itself and shall use a digital signature scheme specified in FCS_COP.1/SIGN to validate the authenticity of the response.

FPT_TUD_EXT.1.2 The OS shall [cryptographically verify] updates to itself using a digital signature prior to installation using schemes specified in FCS_COP.1/SIGN.

FPT_TUD_EXT.2 Trusted Update for Application Software

FPT_TUD_EXT.2.1 The OS shall provide the ability to check for updates to application software and shall use a digital signature scheme specified in FCS_COP.1/SIGN to validate the authenticity of the response.

FPT_TUD_EXT.2.2 The OS shall cryptographically verify the integrity of updates to applications using a digital signature specified by FCS_COP.1/SIGN prior to installation.

5.3.7 TOE Access (FTA)

FTA_TAB.1 Default TOE Access Banners

FTA_TAB.1.1 Before establishing a user session, the **OS** shall display an advisory warning message regarding unauthorized use of the OS.

5.3.8 Trusted Path/Channels (FTP)

FTP_ITC_EXT.1 Trusted Channel Communication

FTP_ITC_EXT.1.1 The OS shall use [

- TLS as conforming to the Functional Package for Transport Layer Security (TLS), version 1.1 as a [client]

.]

and [

- SSH as conforming to the Functional Package for Secure Shell (SSH), version 1.0 as a [client, server]

.]

] to provide a trusted communication channel between itself and authorized IT entities supporting the following capabilities: [*[application initiated TLS, software updates, remote administration via SSH, connections to remote SSH servers]*] that is logically distinct from other communication channels and provides assured identification of its end points and protection of the channel data from disclosure and detection of modification of the channel data.

Application Note: This SFR has been modified by TD0789.

FTP_TRP.1 Trusted Path

FTP_TRP.1.1 The **OS** shall provide a communication path between itself and [remote, local] users that is logically distinct from other communication paths and provides assured identification of its endpoints and protection of the communicated data from [modification, disclosure].

FTP_TRP.1.2 The **OS** shall permit [local users, remote users] to initiate communication via the trusted path.

FTP_TRP.1.3 The **OS** shall require use of the trusted path for [[*all remote administrative actions*]].

5.4 Security Assurance Requirements

27 The TOE assurance requirements for this ST are taken directly from the PP, derived from Common Criteria Version 3.1, Revision 5. The Functional Package for SSH does not define or require any additional SARs. The assurance requirements are summarized in Table 14.

Table 14: Assurance Requirements

Assurance Class	Components	Description
ASE: Security Target	ASE_CCL.1	Conformance Claims
	ASE_ECD.1	Extended Components Definition
	ASE_INT.1	ST Introduction
	ASE_OBJ.2	Security Objectives
	ASE_REQ.2	Derived Security Requirements
	ASE_SPD.1	Security Problem Definition
	ASE_TSS.1	TOE Summary Specification
ADV: Development	ADV_FSP.1	Basic Functional Specification
AGD: Guidance Documentation	AGD_OPE.1	Operational User Guidance
	AGD_PRE.1	Preparative Procedures
ALC: Life-cycle Support	ALC_CMC.1	Labeling of the TOE
	ALC_CMS.1	TOE CM Coverage
	ALC_TSU_EXT.1	Timely Security Updates
ATE: Tests	ATE_IND.1	Independent Testing - Conformance
AVA: Vulnerability Assessment	AVA_VAN.1	Vulnerability Survey

6 TOE Summary Specification

28 The following describes how the TOE fulfills each SFR included in section 5.3.

6.1 Security Audit

6.1.1 FAU_GEN.1

29 The TOE generates audit records that include the following events:

a) Startup and shutdown of audit functions;

b) Successful and failed authentication events;

c) Use of privileged/special rights that involve successful and failed events pertaining to security, audit (/etc/audit/), and configuration changes (sudo/su, pam, sshd, cryptographic policy changes, or changes to trusted databases including passwd and shadow).

d) Successful and unsuccessful privilege/role escalation events including the use of ‘sudo’ and ‘su’ commands and events listed in FAU_GEN.1.1

e) Specifically defined auditable events listed in Table 12

30 The TOE generates and stores audit events locally using the Lightweight Audit Framework (LAF). LAF is designed to serialize and record user space originating events or intercept system calls specified by administrator defined rules. The generated events are placed on the kernel backlog queue until the audit daemon can dequeue and write them to the logs. It has search and reporting utilities to selectively retrieve audit log entries. The framework allows selection of the events to be recorded.

31 Access to the audit logs by normal users is prohibited by the discretionary access control function of the TOE. Access to the audit logs and audit configuration files are allowed only to the system administrator.

32 An audit event consists of one or more lines of text (records) containing fields in a "keyword=value" format. The following information is contained in all audit records:

a) Type: indicates the kind of the information in the record, such as SYSCALL, PATH, USER_LOGIN, or LOGIN

b) Timestamp: Date and time (accurate to the millisecond) that the audit record was generated

c) Serial number: a unique numerical identifier appended to the timestamp to separate events within the same millisecond.

d) Auid (Login ID): the user ID that the user originally authenticated to the system with regardless of the user having changed his real or effective user ID afterwards.

e) Session ID: A unique identifier to disambiguate which login session an event belongs to.

f) Uid: the real user ID of the process at the time the audit event was generated

g) Pid: the process ID of the subject that caused the event.

h) Res: Success or failure results

i) There can be optional information depending on the kind of the event which may include, but is not limited to:

j) The system call that a process made that caused the event

k) The group ID of the subject

l) Hostname or terminal the subject used for performing the operation

m) Information about the intended operation

33 Actions that involve privilege and role escalation to achieve root access can be performed through the use of ‘sudo’ commands.

6.2 Cryptographic Support

34 Table 15 below provides a mapping between the cryptographic SFRs described in the following section and the TOE CAVP certificates.

Table 15: CAVP Mapping

SFR	Algorithm Capability	CAVP
FCS_CKM.1 Cryptographic Key Generation (Refined)	RSA Key Gen (FIPS186-4)	A4771
	ECDSA Key Gen (FIPS186-4)	A4771
	FFC (safe prime) Key Gen (SP 800-56A Rev 3)	Vendor Affirmed
FCS_CKM.2 Cryptographic Key Establishment (Refined)	KAS-ECC-SSC SP800-56Ar3	A4771
	KAS-FFC-SSC SP800-56Ar3	Vendor Affirmed
FCS_CKM_EXT.4 Cryptographic Key Destruction	n/a	~
FCS_COP.1/ENCRYPT Cryptographic Operation - Encryption/Decryption (Refined)	AES-CBC	A4771
	AES-CTR	A4771
	AES-GCM	A4771
FCS_COP.1/HASH Cryptographic Operation - Hashing (Refined)	SHA2-256	A4771
	SHA2-384	A4771
	SHA2-512	A4771 A4770
FCS_COP.1/SIGN Cryptographic Operation - Signing (Refined)	RSA SigGen (FIPS186-4)	A4771
	RSA SigVer (FIPS186-4)	A4771
	ECDSA SigGen (FIPS186-4)	A4771
	ECDSA SigVer (FIPS186-4)	A4771

SFR	Algorithm Capability	CAVP
FCS_COP.1/KEYHMAC Cryptographic Operation - Keyed-Hash Message Authentication (Refined)	HMAC-SHA2-256	A4771
	HMAC-SHA2-384	A4771
	HMAC-SHA2-512	A4771 A4770
FCS_RBG_EXT.1/KCAPI Random Bit Generation (Kernel)	HMAC DRBG (SHA-512)	A4770
FCS_RBG_EXT.1/OSSL Random Bit Generation (OpenSSL)	Counter DRBG (AES256)	A4771
FCS_STO_EXT.1 Storage of Sensitive Data	n/a	~
FCS_SSH_EXT.1 SSH Protocol	n/a	~
FCS_TLS_EXT.1 TLS Protocol	n/a	~

6.2.1 FCS_CKM.1

35 The TOE implements RSA and ECC key generation as specified in FIPS PUB 186-4 “Digital Signature Standard (DSS) Appendix B.3 and Appendix B.4 respectively. The TOE implements FFC key generation using safe primes as specified in NIST SP 800-56A Revision 3. RSA key sizes of 3072 and 4096 are supported. ECC curves P-384 and P-521 are supported. For more detail, please see Table 4. Keys used by the TOE are outlined in the below table:

Table 16: Cryptographic Key Details

Key	Usage	Generator / Initiator	Storage	Destruction / Zeroization
TLS Diffie-Hellman Private Key	Key agreement and key establishment	Generated by the DRBG as specified by FCS_CKM.1 and FCS_CKM.2	Volatile	Overwritten with zeros
TLS Pre-Master Secret	Key agreement and key establishment	Established using Diffie-Hellman	Volatile	Overwritten with zeros
TLS Session Keys	HTTPS/TLS session encryption	Derived from the TLS Pre-Master Secret	Volatile	Overwritten with zeros

Key	Usage	Generator / Initiator	Storage	Destruction / Zeroization
SSH Server Private Key	SSH session authentication	Generated by the DRBG as specified by FCS_CKM.1 or loaded from the filesystem	Non-Volatile (Filesystem API)	Overwritten with pseudo random pattern
SSH User Private Key	SSH session authentication	Generated by the DRBG as specified by FCS_CKM.1 or loaded from the filesystem	Non-Volatile (Filesystem API)	Overwritten with pseudo random pattern
SSH Diffie-Hellman Private Key	Key agreement and key establishment	Generated by the DRBG as specified by FCS_CKM.1 and FCS_CKM.2	Volatile	Removal of power to the memory
SSH Shared Secret	Used for SSH session encryption	Established using Diffie-Hellman	Volatile	Removal of power to the memory
SSH Session Keys	Used for SSH session encryption	Derived from the SSH Shared Secret	Volatile	Removal of power to the memory

6.2.2 FCS_CKM.2

36 For Elliptic curve key establishment, the TOE implements Section 6.1.2.2 of SP 800-56A Rev. 3. The TOE supports Elliptic curve key establishment using the P-384, and P-521 curves.

37 The TOE implements Finite field-based key establishment schemes that generate safe primes which meet NIST SP 800-56A Revision 3.

6.2.3 FCS_CKM_EXT.4

38 For volatile memory, the TOE destroys keys and key material by performing a single overwrite consisting of zeroes. The destruction of SSH keys in volatile memory is achieved via removal of power to the memory. For non-volatile memory, the TOE destroys keys and key material by using the shred utility to perform three (default) overwrites of the logical storage location with a pseudo random pattern. The pseudo random pattern is generated via /dev/urandom. See Table 16 for additional details on key destruction.

6.2.4 FCS_COP.1/ENCRYPT

39 The TOE implements AES in CBC and CTR modes as defined in NIST SP 800-38A, and GCM mode as defined in NIST SP 800-38D. The TOE uses 256-bit key sizes for all modes. See Table 16 for additional detail.

40 The CTR mode counter is a 256-bit value output from the SSH key exchange, so it is guaranteed to be unique. The counter is incremented by 1 for each block that is encrypted. The SSH client rekeys at least every 1 GB of data transmitted using a key, so only a maximum of 2^26 counter values could be used, ensuring the counter does not wrap. The SSH server rekeys at least every 1 GB of data transmitted using a key, so only a maximum of 2^26 counter values could be used, ensuring the counter does not wrap.

6.2.5 FCS_COP.1/HASH

41 The TOE implements SHA-256, SHA-384, and SHA-512 as specified in FIPS PUB 180-4.

6.2.6 FCS_COP.1/SIGN

42 The TOE implements RSA and ECDSA signature generation and verification as specified in FIPS PUB 186-4 Section 4 and Section 5 respectively. RSA key sizes of 2048 (for secure boot only), 3072, and 4096 bits are supported with SHA-256, SHA-384, and SHA-512. ECDSA curves P-384 and P-521 are supported with SHA-256, SHA-384, and SHA-512.

6.2.7 FCS_COP.1/KEYHMAC

43 The TOE implements HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 as specified in FIPS PUB 198-1 and FIPS PUB 180-4.

6.2.8 FCS_RBG_EXT.1/KCAPI

44 The TOE performs deterministic random bit generation using an HMAC_DRBG (SHA-512) in accordance with NIST SP 800-90A. The HMAC_DRBG is seeded with at least 256-bits of entropy from a software-based noise source. This DRBG is used to provide high-security random output for calling applications when invoked using the “getrandom” system call.

6.2.9 FCS_RBG_EXT.1/OSSL

45 The TOE includes the OpenSSL library which implements CTR-DRBG(AES-256) in accordance with NIST SP 800-90A. This DRBG is seeded from KCAPI with at least 256-bits of entropy.

6.2.10 FCS_STO_EXT.1

46 The TOE includes the OpenSSL library to securely store sensitive data. OpenSSL provides file encryption services using AES-256 in CBC and CTR modes using a 256-bit key size.

47 Sensitive data includes application credentials, user passwords, and keys and is stored in the /etc directory. The /etc directory also contains system-wide configuration files and system databases. Access to the files in /etc is limited with strict file permissions and/or encryption and is only accessible to the ‘root’ user and/or the application storing the sensitive data.

48 The TOE also provides the ability to encrypt/decrypt sensitive files using OpenSSL commands.

6.2.11 FCS_SSH_EXT.1, FCS_SSHC_EXT.1, FCS_SSHS_EXT.1

49 The TOE utilizes OpenSSH for its SSHv2 Client and Server implementations. The TOE supports the same algorithms and properties for both implementations:

- Authentication Methods:
  - Public Key
  - Password
- Symmetric Algorithms:
  - aes256-ctr
  - aes256-gcm@openssh.com
- Public Key Algorithms:
  - rsa-sha2-256
  - rsa-sha2-512
  - ecdsa-sha2-nistp384
  - ecdsa-sha2-nistp521
- MACs:
  - hmac-sha2-256
  - hmac-sha2-512
  - implicit
- Key Exchange Methods:
  - diffie-hellman-group16-sha512 (RFC 8268)
  - diffie-hellman-group18-sha512 (RFC 8268)
  - ecdh-sha2-nistp384
  - ecdh-sha2-nistp521

50 The TOE derives cryptographic session keys via shared secret using SSH KDF as defined in RFC 4253 (Section 7.2) and RFC 5656 (Section 4).

51 The TOE drops any SSH packet with a packet_length field greater than 262,144 bytes. The TOE can rekey SSH client connections before a key has been used for over an hour or used to protect more than 1 GB of data. The TOE can also rekey SSH server connections before a key has been used for over an hour or used to protect more than 1 GB of data.

52 OpenSSH utilizes algorithms provided by OpenSSL.

6.2.12 FCS_TLS_EXT.1, FCS_TLSC_EXT.1, FCS_TLSC_EXT.3, FCS_TLSC_EXT.5

53 The TOE provides a TLSv1.2 client implementation with the following ciphersuites:

a) TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5288,

b) TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289,

c) TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289

54 The TOE presents the Supported Groups Extension in the Client Hello message with the secp384r1 and secp521r1 groups.

55 The TOE also presents the signature_algorithms extension by default in the Client Hello message with support for RSA (3072 and 4096) or ECDSA (P-384 and P-521) and the SHA-256, SHA-384, or SHA-512 hash algorithms.

56 The TOE establishes the reference identifier by parsing the DNS Name or IP address for the configured TLS server. The reference identifier is matched against the SAN, if present. If the SAN is not present, the referenced identifier is matched against the CN for DNS. For IP addresses, the TOE matches the identifier against the SAN only. The TOE supports wildcards in the DNS name of the server certificate. The TOE does not support URI reference identifiers, SRV reference identifiers, or certificate pinning.

6.3 User Data Protection (FDP)

6.3.1 FDP_ACF_EXT.1

57 The TOE supports standard UNIX permission bits to provide one form of DAC. There are three sets of three bits that define access for three categories of users: the owning user, users in the owning group, and other users. The three bits in each set indicate the access permissions granted to each user category: one bit for read (r), one for write (w) and one for execute (x). Note that write access to file systems mounted as read only (e. g. CD-ROM) is always rejected (the exceptions are character and block device files which can still be written to as write operations do not modify the information on the storage media). The sticky bit flag is used for world-writable temp directories preventing the removal of files by users other than the owner.

58 Each process has an inheritable “umask” attribute which is used to determine the default access permissions for new objects. It is a bit mask of the user/group/other read/write/execute bits and specifies the access bits to be removed from new objects. For example, setting the umask to “002” ensures that new objects will be writable by the owner and group, but not by others. The umask is defined by the administrator in the /etc/login.defs file or 022 by default if not specified.

59 The TOE also provides support for POSIX type ACLs to define a fine-grained access control on a per-file or per-directory basis. An ACL entry contains the following information:

a) A tag type that specifies the type of the ACL entry

b) A qualifier that specifies an instance of an ACL entry type

c) A permission set that specifies the discretionary access rights for processes identified by the tag type and qualifier

60 An ACL contains exactly one entry of three different tag types (called the "required ACL entries” forming the "minimum ACL"). The standard UNIX file permission bits as described above are represented by the entries in the minimum ACL.

61 A default ACL is an additional ACL which may be associated with a directory. This default ACL has no effect on the access to this directory. Instead, the default ACL is used to initialize the ACL for any file that is created in this directory. If the new file created is a directory it inherits the default ACL from its parent directory. When an object is created within a directory and the ACL is not defined with the function

creating the object, the new object inherits the default ACL of its parent directory as its initial ACL.

62 In addition, the following additional access control bits are processed by the kernel:

a) SUID bit: When an executable marked with the SUID bit is executed, the effective UID of the process is changed to the UID of the owner of the file. The SUID bit for file system objects other than files is ignored.

b) SGID bit: When an executable marked with the SGID bit is executed, the effective GID of the process is changed to the owning GID of the file. The SGID bit for file system objects other than files or directories is ignored.

c) Sticky bit: When a directory is marked with the sticky bit, only the owner of a file system object in that directory can remove it. This bit is commonly used for world-writable directories like /tmp. Only processes with the CAP_FOWNER capability are able to remove the file system object if their UID is different from the owning UID of the file system object.

63 The TOE uses these permissions to protect the following from unauthorized modification:

- Kernel, drivers, and kernel modules – files in:
  - /boot/
  - /usr/lib/modules/
  - /usr/lib/firmware/
- Security audit logs – files in:
  - /var/log/audit/
  - /var/log/secure
- Shared libraries – files in:
  - /usr/lib64/
  - /usr/lib/
- System executables – files in:
  - /usr/sbin/
  - /usr/bin/
  - /usr/libexec/
- System configuration files – files in:
  - /etc/
  - /usr/lib/

64 Both shared libraries and configuration files are stored in /usr/lib/; however, all files in /usr/lib/ are protected from unauthorized modification, regardless of type.

Note: /lib is a symlink to /usr/lib and /lib/64 is a symlink to /usr/lib64. /bin is a symlink to /usr/bin and /sbin is a symlink to /usr/sbin.

65 Additional information on file permissions, umask, and managing access control lists can be found in the ‘Managing File System Permissions’ section of [RHEL] product documentation referenced in Section 2.4.1.

6.4 Identification and Authentication

6.4.1 FIA_AFL.1

66 The TOE will detect when an administrator configurable integer within 1-65,535 unsuccessful authentication attempts for authentication based on username and password occur related to password-based authentication at the local console and over SSH. Once the specified number of unsuccessful authentication attempts for an account has been met, the TOE locks the account.

6.4.2 FIA_UAU.5

67 The TOE supports authentication based on username and password at the local console and over SSH. SSH public key-based authentication is supported over SSH.

68 The TOE performs username and password authentication using a local set of credentials. During password-based login, a PAM (Pluggable Authentication Module) module is invoked which collects the user name and password. The pam_unix module verifies the user is located in the password database and compares a hash of the provided password with one previously stored. If successful, a user session is started. Otherwise, a delay occurs before allowing another attempt if permitted. After login, should the password database indicate that it is time for the user to change the password, they are prompted to do so. Users with expired passwords are prevented from logging in.

69 Users can change their own password as long as it passes administrator defined password complexity rules. Only administrators can add or delete users or change their properties such as group membership.

70 OpenSSH server is able to perform key-based authentication. When a user wants to log on, instead of providing a password, the user sends a signed SSH_MSG_USERAUTH_REQUEST message. If the OpenSSH server can verify the signature using a public key in the user's authorized_keys file, the OpenSSH server considers the user authenticated.

6.4.3 FIA_X509_EXT.1, FIA_X509_EXT.2

71 The TOE uses X.509v3 certificates as defined by RFC 5280 to support authentication for TLS and HTTPS connections.

72 The X.509 certificates are validated using the certificate path validation algorithm defined in RFC 5280, which can be summarized as follows:

a) the public key algorithm and parameters are checked

b) the current date/time is checked against the validity period

c) revocation status is checked using OCSP stapling

d) issuer name of X matches the subject name of X+1

e) extensions are processed

73 The certificate validity check is performed when the TOE receives the certificate during a TLS handshake.

74 When the certificate being validated is for a TLS server, the TOE ensures the Extended Key Usage extension contains the Server Authentication purpose.

75 The TOE ensures all CA certs contain the basic constraints extension and that the CA=TRUE flag is set.

76 The TOE certificate validation algorithm also ensures that the certificate path terminates in a trusted root CA (i.e., a CA certificate configured on the TOE as trusted).

77 If the validity check of a certificate fails, or the OCSP server cannot otherwise provide a valid or current response, the certificate is rejected.

6.5 Security Management

6.5.1 FMT_MOF_EXT.1

78 The TOE restricts all “Administrator” management activities listed in FMT_SMF_EXT.1.1 to users who are members of the “wheel” group. Members of this group are considered the administrators, because group membership allows users to elevate their privileges, allowing management of the TOE, using the sudo command.

6.5.2 FMT_SMF_EXT.1

79 The TOE allows the administrators to perform all of the management activities, indicated with the marking of 'X', in Section 5.3.5 Table 13. Management activities are restricted to authorized administrators only and therefore, non-administrative users are not allowed to manage the TOE.

80 Administrators are assigned to specific groups that control the privileges required to access and manage these functions.

6.6 Protection of the TSF

6.6.1 FPT_ACF_EXT.1

81 The TOE uses groups and ACLs to assign the file and directory permissions described in FDP_ACF_EXT.1 to control access and prevent unprivileged or non-root users from modifying:

a) Kernel and its drivers/modules

b) Security audit logs

c) Shared libraries

d) System executables

e) System configuration files

6.6.2 FPT_ASLR_EXT.1/Xeon, FPT_ASLR_EXT.1/z16, FPT_ASLR_EXT.1/Power10

82 On Intel x86-64 processors, the TOE executables are compiled as Position Independent Executables with the following amount of randomization:

a) exec 30 bits

b) heap 30 bits

c) so 29 bits

d) mmap 29 bits

e) stack 30 bits

83 On IBM z16 processors, the TOE executables are compiled as Position Independent Executables with the following amount of randomization:

a) exec 11 bits

b) heap 19 bits

c) so 13 bits

d) mmap 13 bits

e) stack 21 bits

84 On Power10 processors, the TOE executables are compiled as Position Independent Executables with the following amount of randomization:

a) exec 14 bits

b) heap 14 bits

c) so 16 bits

d) mmap 16 bits

e) stack 26 bits

85 On all architectures, the TOE developer guidance instructs developers to compile non-TOE executables with PIE flags, so non-TOE executables have the same amount of randomization.

6.6.3 FPT_SBOP_EXT.1

86 The TOE is compiled using gcc with the option “stack-protector-strong”. This option adds a stack canary and associated verification code during the entry and exit of function frames (those that have local array definitions, have references to local frame addresses or functions with vulnerable objects) to detect stack-based buffer overflows. Only allocated variables on the stack are considered.

87 The gcc compiler is used on all platforms listed in Table 5.

6.6.4 FPT_SRP_EXT.1

88 The TOE restricts execution of programs to those allowed based on file path and hash. The file paths and hashes of allowed applications are added to the trust database as part of the installation process for each application.

6.6.5 FPT_TST_EXT.1

89 The boot chain broadly consists of the following steps:

a) Platform responsibility:

   i) Firmware / Initialization

   ii) First Stage Boot Loader

b) TOE responsibility:

   i) Second Stage Boot Loader

   ii) Linux Kernel

90 Each link in the boot chain verifies the next link until execution is handed over to the Linux Kernel.

91 Platform specific mechanisms are as follows:

a) **Intel x86-64 UEFI platforms.** UEFI Secure Boot is used as follows:

i) UEFI firmware stores the hashes (SHA256) and public keys (RSA 2048) for trusted loaders and EFI applications (Allow DB)

ii) UEFI verifies the first stage boot loader (shim.efi) digital signature against the Allow DB (only proceeding if the verification succeeds)

iii) Shim.efi holds its own databases of trusted public keys (RSA 2048) and code hashes (SHA256) that are allowed to be loaded. This is used to verify the signature of the second-stage boot loader, GRUB 2 (grubx64.efi). Execution proceeds only if the verification succeeds.

iv) Finally, GRUB 2 uses the RSA 2048 code signing key to verify the SHA256 signature on the OS kernel before passing control to the kernel.

b) **IBM z16 platforms.** Linux Secure Boot based on List Directed Initial Program Load (LD-IPL) processing is used as follows:

i) List Directed Initial Program Load locates the signature data stored on disk.

ii) The machine loader locates the address of the stage 3 bootloader to be booted and verifies that its signature matches the certificate, only proceeding if the verification succeeds.

iii) The machine loader then turns control over to the stage 3 bootloader (a component of the larger zipl bootloader) which reads the SHA256 signature and RSA 2048 public key for the kernel from the IPL Parameter Block and verifies the signature of the kernel. If the signature verification succeeds, the stage 3 bootloader passes control to the kernel. If there is no signature, or signature verification fails, the stage 3 bootloader terminates the boot.

c) **Power10 platforms.**

i) Each component of the firmware stack, including hostboot, the POWER Hypervisor, and partition firmware, is signed by the platform manufacturer and verified as part of the Initial Program Load process.

ii) Partition Firmware verifies GRUB 2 using the SHA256 signature and RSA 2048 public key.

iii) Finally, GRUB 2 uses the code signing key RSA 2048 to verify the SHA256 hash signature on the OS kernel before passing control to the kernel.

92 On all platforms, the kernel has additional embedded keys that are used to authenticate drivers and kernel modules.

6.6.6 FPT_TUD_EXT.1, FPT_TUD_EXT.2

93 The TOE has the ability to check for updates to itself and application software. Both types of updates are verified by RSA 4096 with SHA-256 prior to installation. Updates to the TOE and application software are downloaded by the TOE from an organization’s local repository.

6.7 TOE Access

6.7.1 FTA_TAB.1

94 The TOE can be configured to display an administrator configured advisory warning message prior to establishing a local or remote interactive user session.

6.8 Trusted Path/Channels

6.8.1 FTP_ITC_EXT.1

95 The TOE provides a TLS Client protocol implementation which allows applications to protect communications with remote IT entities and perform secure software updates via the local repository. The TOE uses the SSH server protocol to protect communications with remote users. The TOE also allows users to securely connect to remote servers using SSH.

6.8.2 FTP_TRP.1

96 The TOE provides a trusted path with local and remote users. The TOE uses the SSH Server protocol to protect the communications with remote users.

6.9 Stack Smashing Protection

97 The TOE includes several binaries that were not compiled with stack-smashing protections enabled for a number of reasons. The reasons are listed below, followed by a list of binaries to which that reason applies.

98 glibc has special code for stack unwinding, exception handling, and other handwritten assembler. As such, it cannot enable the compiler-based stack protector.

a) /usr/lib64/libc.so.6

b) /usr/lib64/ld-linux-x86-64.so.2 (x86_64)

c) /usr/lib/ld64.so.1 (IBM Z)

d) /usr/lib64/ld64.so.2 (Power10)

e) /usr/sbin/ldconfig

99 The following binaries do not have stack smashing protection enabled because there is no compiled C/C++ code:

a) /usr/sbin/grub2-set-bootflag (x86_64, Power10)

b) /usr/lib64/libefivar.so.1.38 (x86_64)

c) /usr/lib64/libefiboot.so.1.38 (x86_64)

d) /usr/lib/modules/5\.14\.0-.*/vdso/vdso(|32|64)\.so (x86_64, IBM Z)

e) /usr/sbin/thin_metadata_unpack

f) /usr/sbin/thin_metadata_pack

g) /usr/lib/udev/prefixdevname

h) /usr/lib64/libpython3.so

i) /usr/lib64/libicudata.so.67.1

100 The following binaries contain indirect functions (ifunc) so they cannot enable the compiler-based stack protector:

a) /usr/lib64/gconv/.*\.so (IBM Z)

b) /usr/lib64/libm\.so\.6 (x86_64, Power10)

c) /usr/lib64/libmvec\.so\.1 (x86_64)

101 The newns application is a simple wrapper to create a new mount namespace. It only has the argc/argv variables in main(). The program calls unshare(CLONE_NEWNS) to create the new namespace. It then passes argv to execvp. Under normal execution, the call to execvp doesn't return. If execvp fails, it calls the exit syscall which also does not return. In either case, the stack's return address is not used.

a) /usr/libexec/os-prober/newns

6.10 Timely Security Updates

6.10.1 ALC_TSU_EXT.1

102 Red Hat accepts reports of security issues at the secalert@redhat.com email address. Red Hat provides a public GPG key (available at https://access.redhat.com/security/team/contact) so the reporter can protect sensitive aspects of a report. Email sent to secalert@redhat.com is read and acknowledged with a non-automated response within three working days. For issues that are complicated and require significant attention, Red Hat will open an investigation and will provide reporters with a mechanism to check the status at any time.

103 For security issues under embargo, Red Hat does not disclose, discuss, or confirm security issues until an investigation is conducted and the vulnerability is made public. Once an embargoed issue has been made public, Red Hat publishes documentation regarding the flaw including technical details on the issue, a Common Vulnerabilities and Exposures (CVE) identifier, a Common Vulnerabilities Security Score (CVSS), a Red Hat Severity Rating, and the Red Hat products impacted by the vulnerability.

104 Red Hat distributes information about security issues in its products through 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.

105 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.

7 Rationale

7.1 Conformance Claim Rationale

106 The following rationale is presented with regard to the PP conformance claims:

a) **TOE type.** As identified in section 2.1, the TOE is a general purpose operating system, consistent with the PP_OS_v4.3 in addition to PKG_SSH_v1.0, and PKG_TLS_v1.1.

b) **Security problem definition.** As shown in section 3, the threats, OSPs and assumptions are reproduced directly from the PP_OS_v4.3.

c) **Security objectives.** As shown in section 4, the security objectives are reproduced directly from the PP_OS_v4.3.

d) **Security requirements.** As shown in section 5, the security requirements are reproduced directly from the PP_OS_v4.3, PKG_SSH_v1.0, and PKG_TLS_v1.1. No additional requirements have been specified.

7.2 Security Requirements Rationale

107 All security requirements are drawn directly from the PP_OS_v4.3.

108 Table 10 presents a mapping between threats and security objectives.

109 The below table presents a mapping between objectives and SFRs and is reproduced from the PP_OS_v4.3.

Table 17: Assurance Requirements

Objective	Addressed By	Rationale
O.ACCOUNTABILITY	FAU_GEN.1	Supports the objective by requiring that critical event information be gathered by the TOE.
	FTP_ITC_EXT.1	Supports the objective by ensuring that audit information can be securely transmitted to remote systems for analysis.
O.INTEGRITY	FPT_SBOP_EXT.1 FPT_ASLR_EXT.1	Supports the objective by requiring that OS applications be hardened against buffer overflow attacks
	FPT_TUD_EXT.1	Supports the objective by requiring that the OS be able to check for critical updates.
	FPT_TUD_EXT.2	Supports the objective by requiring that the OS verify updates before applying them.
	FCS_COP.1/HASH	Supports the objective by requiring the TSF to implement hash algorithms that

Objective	Addressed By	Rationale
		are used in support of protected communications.
	FCS_COP.1/SIGN	Supports the objective by requiring the TSF to implement digital signature algorithms that are used in support of protected communications.
	FCS_COP.1/KEYHMAC	Supports the objective by requiring the TSF to implement HMAC algorithms that are used in support of protected communications.
	FPT_ACF_EXT.1	Supports the objective by requiring the TSF restrict unprivileged users from changing critical components.
	FPT_SRP_EXT.1	Supports the objective by requiring the TSF to implement a configurable allowlist mechanism.
	FIA_X509_EXT.1	Supports the objective by requiring the TSF to validate certificates using industry standards.
	FPT_TST_EXT.1	Supports the objective by requiring the TSF to verify executable code critical to its operation.
	FTP_ITC_EXT.1	Supports the objective by requiring the OS to provide a trusted channel for critical communication.
	FIA_AFL.1	Supports the objective by requiring the TSF to respond accordingly when the number of failed authentication attempts reaches a specified threshold.
	FIA_UAU.5	Supports the objective by requiring the OS to provide standard authentication mechanisms.
O.MANAGEMENT	FMT_MOF_EXT.1	Supports this objective by requiring the TOE to restrict the ability to perform certain management functions to a privileged user.
	FMT_SMF_EXT.1	Supports this objective by requiring the TOE to implement specific management functions.

Objective	Addressed By	Rationale
	FTA_TAB.1	Supports this objective by requiring the TOE to implement a trusted path between itself and users.
	FTP_TRP.1	Supports this objective by requiring a trusted path between users and the OS.
O.PROTECTED_ STORAGE	FCS_STO_EXT.1	Supports this objective by requiring the OS to provide encrypted storage.
	FCS_RBG_EXT.1	Supports this objective by requiring the OS to generate random bits according to industry standards.
	FCS_COP.1/ENCRYPT	Supports this objective by requiring the OS to perform encryption according to industry standards.
	FDP_ACF_EXT.1	Supports this objective by requiring the OS to implement access controls.
O.PROTECTED_ COMMS	FCS_RBG_EXT.1	Supports this objective by requiring the OS to generate random bits according to industry standards.
	FCS_CKM.1	Supports this objective by requiring the TSF to generate asymmetric cryptographic keys to industry standards.
	FCS_CKM.2	Supports this objective by requiring the TSF to perform key establishment according to industry standards.
	FCS_CKM_EXT.4	Supports this objective by requiring the TSF to destroy key material according to industry standards.
	FCS_COP.1/ENCRYPT	Supports this objective by requiring the TSF to encrypt data according to industry standards
	FCS_COP.1/HASH	Supports this objective by requiring the TSF to hash data according to industry standards.
	FCS_COP.1/SIGN	Supports this objective by requiring the TSF to cryptographically sign data according to industry standards.

Objective	Addressed By	Rationale
	FCS_COP.1/KEYHMAC	Supports this objective by requiring the TSF to perform keyed hashes according to industry standards.
	FDP_IFC_EXT.1	Supports this objective by requiring the TSF to be compatible with at least one VPN.
	FIA_X509_EXT.1	Supports the objective by requiring the TSF to validate certificates using industry standards.
	FIA_X509_EXT.2	Supports this objective by requiring the TSF to validate TLS and related encrypted connections with x509 certificates.
	FTP_ITC_EXT.1	Supports the objective by requiring the OS to provide a trusted channel for critical communication.
	FCS_TLS_EXT.1 (TLS Package)	FCS_TLS_EXT.1 supports the objective by defining the TOE's implementation of TLS and DTLS if this protocol is used for protected communications.
	FCS_TLSC_EXT.1 (TLS Package)	FCS_TLSC_EXT.1 supports the objective by defining the TOE's implementation of TLS as a client for protected communications.
	FCS_TLSC_EXT.3 (TLS Package) (Objective)	FCS_TLSC_EXT.3 supports the objective by requiring the TSF to support the TLS signature algorithms extension as part of establishing TLS protected communications.
	FCS_TLSC_EXT.5 (TLS Package)	FCS_TLSC_EXT.5 supports the objective be defining the TOE's implementation of supported groups extension for TLS as a client for protected communications.
	FCS_SSH_EXT.1 (SSH Package)	FCS_SSH_EXT.1 supports the objective by defining the TOE's implementation of SSH if this protocol is used for protected communications.
	FCS_SSHC_EXT.1 (SSH Package)	FCS_SSHC_EXT.1 supports the objective by defining the TOE's implementation of SSH as a client if this

Objective	Addressed By	Rationale
		protocol is used for protected communications.
	FCS_SSHS_EXT.1 (SSH Package)	FCS_SSHS_EXT.1 supports the objective by defining the TOE's implementation of SSH as a server if this protocol is used for protected communications.

110 **NOTE:** Table 17 has been modified by TD0713 from its original presentation in PP_OS_v4.3. The table has also been modified to reflect the claimed SFRs.