Red Hat Enterprise Linux 8.1 Security Target

Acumen Security, LLC.

Document Version: 0.6

Table Of Contents

Revision History

Version	Date	Description
0.1	August 2019	Initial Draft
0.2	October 2019	Updated based on TDs and review
0.3	May 2020	Updated
0.4	September 2020	Updated
0.5	October 2020	Updated for check-out
0.6	December 2020	Update for ECR comments

1 Security Target Introduction

1.1 Security Target and TOE Reference

This section provides information needed to identify and control this ST and its TOE.

Category	Identifier
ST Title	Red Hat Enterprise Linux 8.1 Security Target
ST Version	0.6
ST Date	December 2020
ST Author	Acumen Security, LLC.
TOE Identifier	Red Hat Enterprise Linux
TOE Software Version	8.1
TOE Developer	Red Hat, Inc.
Key Words	Operating System, SSH, TLS, Linux

Table 1 TOE/ST Identification

1.2 TOE Overview

Red Hat® Enterprise Linux® is the world's leading enterprise Linux platform. It's an open source operating system (OS) that supports multiple users, user permissions, access controls, and cryptographic functionality.

1.3 TOE Architecture

1.3.1 TOE Evaluated Configuration

The TOE also supports (sometimes optionally) secure connectivity with several other IT environment devices as described in Table 2 below:

Component	Required	Usage/Purpose Description for TOE performance
Workstation with SSH Client	No	This includes any IT Environment Management workstation with an SSH client installed that is used by the TOE users (including administrators) to remotely connect to the TOE through SSH protected channels. Any SSH client that supports SSHv2 may be used.
Audit Server	No	The audit server is used for remote storage of audit records that have been generated by and transmitted from the TOE.
Update Server	Yes	Provides the ability to check for updates to the TOE as well as providing signed updates.

Table 2 IT Environment Components

1.3.2 Physical Boundaries

The TOE itself does not have physical boundaries; however, the TOE was evaluated on the following hardware:

Vendor	Model	CPU
Dell Inc.	PowerEdge R440	Xeon Silver 42xx
Dell Inc.	PowerEdge R540	Xeon Silver 42xx
Dell Inc.	PowerEdge R640	Xeon Silver 42xx
Dell Inc.	PowerEdge R740	Xeon Silver 42xx
Dell Inc.	PowerEdge R740XD	Xeon Silver 42xx
Dell Inc.	PowerEdge 840	Xeon Silver 42xx
Dell Inc.	PowerEdge 940	Xeon Silver 42xx

Vendor	Model	CPU
Dell Inc.	PowerEdge 940sa	Xeon Silver 42xx

Table 3 Evaluated Hardware

The Xeon Silver 4200 series processors are 2nd Generation Intel® Xeon® Scalable Processors and implement the Cascade Lake microarchitecture.

The TOE was tested on a PowerEdge R740 with a Xeon Silver 4216 CPU.

1.3.3 Security Functions provided by the TOE

The TOE provides the security functionality required by [GPOSPP] and [SSHEP].

1.3.3.1 Security Audit

The TOE generates and stores audit events using the Lightweight Audit Framework (LAF). The LAF is designed to be an audit system making Linux compliant with the requirements from Common Criteria by intercepting all system calls and retrieving audit log entries from privileged user space applications. The framework allows configuring the events to be recorded from the set of all events that are possible to be audited. Each audit record contains the date and time of event, type of event, subject identity, user identity, and result (success/fail) of the action if applicable.

1.3.3.2 Cryptographic Support

The TOE provides a broad range of cryptographic support; providing SSHv2 and TLSv1.2 protocol implementations in addition to individual cryptographic algorithms.

The cryptographic services provided by the TOE are described below.

Cryptographic Protocol	Use within the TOE
SSH Client	The TOE allows administrators and users to connect to remote SSH servers.
SSH Server	The TOE allows remote administrators to connect using SSH.
TLS Client	The TOE connects to remote trusted IT entities using TLS.

Table 4 TOE Cryptographic Protocols

The TOE includes four cryptographic libraries/implementations. Each of these cryptographic algorithms have been validated for conformance to the requirements specified in their respective standards, as identified below.

Algorithm	Related SFRs	TOE Use	CAVP Certificate #
OpenSSL v1.1.1c with algorithm version rhel8.20190624cc			
AES	FCS_COP.1(1) FCS_COP.1(1)/SSH FCS_SSHC_EXT.1 FCS_SSHS_EXT.1 FCS_TLSC_EXT.1 FCS_STO_EXT.1	SSH AES CBC and CTR modes with 128 and 256-bit keys TLS AES CBC and GCM modes with 128 and 256-bit keys File Encryption using AES CBC with 128 and 256-bit keys	A796
Diffie-Hellman	FCS_CKM.2 FCS_SSHC_EXT.1 FCS_SSHS_EXT.1 FCS_TLSC_EXT.1	SSH Diffie-Hellman Group 14 Key Establishment TLS Diffie-Hellman Group 14 Key Establishment	N/A
DRBG	FCS_DRBG_EXT.1	CTR_DRBG (AES-256)	A796

Algorithm	Related SFRs	TOE Use	CAVP Certificate #
ECDSA	FCS_CKM.1 FCS_COP.1(3) FCS_SSHC_EXT.1 FCS_SSHS_EXT.1 FCS_TLSC_EXT.1 FCS_TLSC_EXT.2 FCS_TLSC_EXT.4	SSH ECDSA P-256 and P-384 Host Key and User Key Generation SSH EC Diffie-Hellman P-256, P-384, and P-521 Key Generation SSH ECDSA P-256 and P-384 Host and User Signature Generation and Verification TLS ECDSA P-256, P-384, and P-521 Client Key Generation TLS EC Diffie-Hellman P-256, P-384, and P-521 Key Generation TLS ECDSA P-256, P-384, and P-521 Signature Generation and Verification	A796
HMAC	FCS_COP.1(4) FCS_SSHC_EXT.1 FCS_SSHS_EXT.1 FCS_TLSC_EXT.1	SSH HMAC-SHA-256 and HMAC-SHA-512 TLS HMAC-SHA-1, HMAC-SHA-256, and HMAC-SHA-384 TLS HMAC-SHA-256 and HMAC-SHA-384 Key Derivation	A796
KAS	FCS_CKM.2 FCS_SSHC_EXT.1 FCS_SSHS_EXT.1 FCS_TLSC_EXT.2	SSH EC Diffie-Hellman P-256, P-384, and P-521 Key Establishment TLS EC Diffie-Hellman P-256, P-384, and P-521 Key Establishment	A796
RSA	FCS_CKM.1 FCS_CKM.2 FCS_COP.1(3) FCS_SSHC_EXT.1 FCS_SSHS_EXT.1 FCS_TLSC_EXT.1 FCS_TLSC_EXT.4 FPT_TST_EXT.1 FPT_TUD_EXT.1 FPT_TUD_EXT.2	SSH RSA 2048-bit, 3072-bit, and 4096-bit Host Key and User Key Generation SSH RSA 2048-bit, 3072-bit, and 4096-bit Host and User Signature Generation and Verification TLS RSA 2048-bit, 3072-bit, and 4096-bit Client Key Generation TLS RSA 2048-bit, 3072-bit, and 4096-bit Key Establishment (CAVP certificate is N/A) TLS RSA 2048-bit, 3072-bit, and 4096-bit Signature Generation and Verification Self-Test RSA 2048 Signature Verification Trusted Update RSA 4096 Signature Verification	A796
SHS	FCS_COP.1(2) FCS_SSHC_EXT.1 FCS_SSHS_EXT.1	SSH SHA-1, SHA-256, SHA-384, and SHA-512 Key Derivation SHA-1, SHA-256, SHA-384, and SHA-512 for Digital Signatures and HMACs	A796

Table 5 CAVP Algorithm Testing References

The OpenSSL library provides the TLS Client function. The OpenSSL library also provides the cryptographic algorithms for the SSH Client, SSH Server, trusted update, and secure boot security functions.

1.3.3.3 User Data Protection

Discretionary Access Control (DAC) allows the TOE to assign owners to file system objects and Inter-Process Communication (IPC) objects. The owners are allowed to modify Unix-type permission bits for these objects to permit or deny access for other users or groups. The DAC mechanism also ensures that untrusted users cannot tamper with the TOE mechanisms.

The TOE also implements POSIX Access Control Lists (ACLs) that allow the specification of the access to individual file system objects down to the granularity of a single user.

1.3.3.4 Identification and Authentication

User identification and authentication in the TOE includes all forms of interactive login (e.g. using the SSH protocol or log in at the local console) as well as identity changes through the su or sudo command. These all rely on explicit authentication information provided interactively by a user.

The authentication security function allows password-based authentication. For SSH access, public-key-based authentication is also supported.

Password quality enforcement mechanisms are offered by the TOE which are enforced at the time when the password is changed.

1.3.3.5 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.

1.3.3.6 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 self-protection mechanisms are implemented and enforced:

- Address Space Layout Randomization for user space code.
- Stack buffer overflow protection using stack canaries.
- Secure Boot ensuring that the boot chain up to and including the kernel together with the boot image (initramfs) is not tampered with.
- Updates to the operating system are only installed after their signatures have been successfully validated.
- Application Whitelisting restricts execution to known/trusted applications.

1.3.3.7 TOE Access

The TOE displays informative banners before users are allowed to establish a session.

1.3.3.8 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.

1.3.4 TOE Documentation

- [ST] Red Hat Enterprise Linux 8.1 Security Target, Version 0.6
- [AGD] Red Hat Enterprise Linux 8.1 CC Guidance, Version 1.4

1.3.5 Other References

- Protection Profile for General Purpose Operating Systems, Version 4.2.1 [GPOSPP]
- Extended Package for Secure Shell (SSH), Version 1.0 [SSHEP]

2 Conformance Claims

2.1 CC Conformance

This TOE is conformant to:

- Common Criteria for Information Technology Security Evaluations Part 1, Version 3.1, Revision 5, April 2017
- Common Criteria for Information Technology Security Evaluations Part 2, Version 3.1, Revision 5, April 2017: Part 2 extended
- Common Criteria for Information Technology Security Evaluations Part 3, Version 3.1, Revision 5, April 2017: Part 3 extended

2.2 Protection Profile Conformance

This TOE is conformant to:

- Protection Profile for General Purpose Operating Systems, Version 4.2.1 [GPOSPP]
- Extended Package for Secure Shell (SSH), Version 1.0 [SSHEP]

2.3 Conformance Rationale

This Security Target provides exact conformance to the [GPOSPP] and [SSHEP]. The security problem definition, security objectives and security requirements in this Security Target are all taken from the Protection Profile performing only operations defined there.

2.3.1 Technical Decisions

All NIAP Technical Decisions (TDs) issued to date that are applicable to the [GPOSPP] and [SSHEP] have been considered. The following table identifies all applicable TDs:

Identifier	Applicable	Exclusion Rationale (if applicable)
TD0525 – Updates to Certificate Revocation (FIA_X509_EXT.1)	Yes	
TD0501 – Cryptographic selections and updates for OS PP	Yes	
TD0496 – GPOS PP adds allow-with statement for VPN Client V2.1	Yes	
TD0493 – X.509v3 certificates when using digital signatures for Boot Integrity	Yes	
TD0463 – Clarification for FPT_TUD_EXT	Yes	
TD0441 – Updated TLS Ciphersuites for OS PP	Yes	
TD0386 – Platform-Provided Verification of Update	Yes	
TD0365 – FCS_CKM_EXT.4 selections	Yes	

Table 6 GPOS Technical Decisions

Identifier	Applicable	Exclusion Rationale (if applicable)
TD0446 – Missing selections for SSH	Yes	
TD0420 – Conflict in FCS_SSHC_EXT.1.1 and FCS_SSHS_EXT.1.1	Yes	
TD0332 – Support for RSA SHA2 host keys	Yes	
TD0331 – SSH Rekey Testing	Yes	
TD0240 – FCS_COP.1.1(1) Platform provided crypto for encryption/decryption	Yes	

Table 7 SSH EP Technical Decisions

3 Security Problem Definition

The security problem definition has been taken from the [GPOSPP]. It is reproduced here for the convenience of the reader. The security problem is described in terms of the threats that the TOE is expected to address, assumptions about the operational environment, and any organizational security policies that the TOE is expected to enforce.

3.1 Threats

The following threats are drawn directly from the [GPOSPP].

ID	Threat
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.

Table 8 Threats

3.2 Assumptions

The following assumptions are drawn directly from the [GPOSPP].

ID	Assumption
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.

Table 9 Assumptions

3.3 Organizational Security Policies

The [GPOSPP] and [SSHEP] do not define any OSPs.

4 Security Objectives

The security objectives have been taken from the [GPOSPP]. They are reproduced here for the convenience of the reader.

4.1 Security Objectives for the TOE

The following security objectives for the operational environment assist the TOE in correctly providing its security functionality. These track with the assumptions about the environment.

ID	Objective for the TOE
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.

Table 10 Objectives for the TOE

4.2 Security Objectives for the Operational Environment

The following security objectives for the operational environment assist the TOE in correctly providing its security functionality. These track with the assumptions about the environment.

ID	Objective for the Operation Environment
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.

Table 11 Objectives for the Operational Environment

5 Security Requirements

This section identifies the Security Functional Requirements for the TOE. The Security Functional Requirements included in this section are derived from Part 2 of the Common Criteria for Information Technology Security Evaluation, Version 3.1, Revision 5, dated: April 2017 and all international interpretations.

Requirement	Description
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(1)	Cryptographic Operation - Encryption/Decryption (Refined)
FCS_COP.1(1)/SSH	Cryptographic Operation - Encryption/Decryption (Refined)
FCS_COP.1(2)	Cryptographic Operation - Hashing (Refined)
FCS_COP.1(3)	Cryptographic Operation - Signing (Refined)
FCS_COP.1(4)	Cryptographic Operation - Keyed-Hash Message Authentication (Refined)
FCS_RBG_EXT.1	Random Bit Generation
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_TLSC_EXT.1	TLS Client Protocol
FCS_TLSC_EXT.2	TLS Client Protocol
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	Address Space Layout Randomization
FPT_SBOP_EXT.1	Stack Buffer Overflow Protection
FPT_SRP_EXT.1	Software Restriction Policies
FPT_TST_EXT.1	Boot Integrity
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

Table 12 SFRs

5.1 Conventions

The CC defines operations on Security Functional Requirements: assignments, selections, assignments within selections and refinements. This document uses the following font conventions to identify the operations defined by the CC:

- Assignment: Indicated with *italicized* text;
- Refinement: Indicated with **bold** text;
- Selection: Indicated with underlined text;
- Iteration: Indicated by appending the SFR name with a slash and unique identifier suggesting the purpose of the iteration, e.g. '/SSH for an SFR relating to SSH functionality and/or a sequential number in parentheses, e.g. (1).
- Where operations were completed in the PP, EP, or Module itself; the formatting used in the PP, EP, or Module has been retained.

Extended SFRs are identified by having a label 'EXT' after the requirement name. Formatting conventions outside of operations matches the formatting specified within the PP, EP, or Module.

5.2 Security Functional Requirements

5.2.1 Security Audit (FAU)

5.2.1.1 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),*
  - *System reboot, restart, and shutdown events (Success/Failure),*
  - *Kernel module loading and unloading events (Success/Failure),*
  - *Administrator or root-level access events (Success/Failure).*

  ]

].

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 the 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, **[***no other information***]**

5.2.2 Cryptographic Support (FCS)

5.2.2.1 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 2048-bit or greater that meet the following: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.3 ,***
- ***ECC schemes using "NIST curves" P-256, 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]**

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

5.2.2.2 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: [

- ***RSA-based key establishment schemes that meets the following: RSAES-PKCS1-v1_5 as specified in Section 7.2 of RFC 8017, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.2,***
- ***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",***
- ***Finite field-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",***

] ~~that meets the following: [assignment: list of standards]~~ .

5.2.2.3 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],*

  ],
- *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.

5.2.2.4 FCS_COP.1(1) Cryptographic Operation - Encryption/Decryption (Refined)

FCS_COP.1.1(1)

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)***

] and [

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

] and cryptographic key sizes [*128-bit, 256-bit*] ~~that meet the following: [assignment: list of standards]~~.

5.2.2.5 FCS_COP.1(1)/SSH Cryptographic Operation - Encryption/Decryption (Refined)

FCS_COP.1.1(1)/SSH

The SSH software shall [*invoke platform-provided*] encryption/decryption services for data in accordance with a specified cryptographic algorithm AES-CTR (as defined in NIST SP 800-38A) mode and cryptographic key sizes [*128-bit, 256-bit*].

5.2.2.6 FCS_COP.1(2) Cryptographic Operation - Hashing (Refined)

FCS_COP.1.1(2)

The **OS** shall perform [*cryptographic hashing services*] in accordance with a specified cryptographic algorithm [*SHA-1 and [*

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

]] **and message digest sizes 160 bits and [**

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

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

5.2.2.7 FCS_COP.1(3) Cryptographic Operation - Signing (Refined)

FCS_COP.1.1(3)

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 or greater that meet the following: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Section 4,***
- ***ECDSA schemes using "NIST curves" P-256, 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]~~.

5.2.2.8 FCS_COP.1(4) Cryptographic Operation - Keyed-Hash Message Authentication (Refined)

FCS_COP.1.1(4)

The **OS** shall perform [*keyed-hash message authentication services*] in accordance with a specified cryptographic algorithm **[*SHA-1, SHA-256, SHA-384, SHA-512*] with key sizes [*160 bits, 256 bits, 384 bits, 512 bits*] and message digest** sizes [*160 bits, 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*].

5.2.2.9 FCS_RBG_EXT.1 Random Bit Generation

FCS_RBG_EXT.1.1

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

The deterministic RBG used by the OS shall be seeded by an entropy source that accumulates entropy from a [

- *platform-based noise source*

] with a minimum of [

- *256 bits*

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

5.2.2.10 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.

5.2.2.11 FCS_SSH_EXT.1 SSH Protocol

FCS_SSH_EXT.1.1

The SSH software shall implement the SSH protocol that complies with RFCs 4251, 4252, 4253, 4254 and [*5656, 6668*] as a [*client, server*]

5.2.2.12 FCS_SSHC_EXT.1 SSH Protocol - Client

FCS_SSHC_EXT.1.1

The SSH client shall ensure that the SSH protocol implementation supports the following authentication methods as described in RFC 4252: public key-based, and [*password-based*].

FCS_SSHC_EXT.1.2

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

FCS_SSHC_EXT.1.3

The SSH software shall ensure that the SSH transport implementation uses the following encryption algorithms and rejects all other encryption algorithms: aes128-ctr, aes256-ctr, [*aes128-cbc, aes256-cbc*].

FCS_SSHC_EXT.1.4

The SSH client shall ensure that the SSH transport implementation uses [*ssh-rsa, rsa-sha2-256, rsa-sha2-512, ecdsa-sha2-nistp256*] and [*ecdsa-sha2-nistp384*] as its public key algorithm(s) and rejects all other public key algorithms.

FCS_SSHC_EXT.1.5

The SSH client shall ensure that the SSH transport implementation uses [*hmac-sha2-256, hmac-sha2-512*] and [*no other MAC algorithms*] as its MAC algorithm(s) and rejects all other MAC algorithm(s).

FCS_SSHC_EXT.1.6

The SSH client shall ensure that [*diffie-hellman-group14-sha1, ecdh-sha2-nistp256*] and [*ecdh-sha2-nistp384, ecdh-sha2-nistp521*] are the only allowed key exchange methods used for the SSH protocol.

FCS_SSHC_EXT.1.7

The SSH server shall ensure that the SSH connection be rekeyed after [*no more than 1 Gigabyte of data has been transmitted, no more than 1 hour*] using that key.

FCS_SSHC_EXT.1.8

The SSH client shall ensure that the SSH client authenticates the identity of the SSH server using a local database associating each host name with its corresponding public key or [*no other methods*] as described in RFC 4251 section 4.1.

5.2.2.13 FCS_SSHS_EXT.1 SSH Protocol - Server

FCS_SSHS_EXT.1.1

The SSH server shall ensure that the SSH protocol implementation supports the following authentication methods as described in RFC 4252: public key-based, and [*password-based*].

FCS_SSHS_EXT.1.2

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

FCS_SSHS_EXT.1.3

The SSH server shall ensure that the SSH transport implementation uses the following encryption algorithms and rejects all other encryption algorithms: aes128-ctr, aes256-ctr, [*aes128-cbc, aes256-cbc*].

FCS_SSHS_EXT.1.4

The SSH server shall ensure that the SSH transport implementation uses [*ssh-rsa, rsa-sha2-256, rsa-sha2-512, ecdsa-sha2-nistp256*] and [*ecdsa-sha2-nistp384*] as its public key algorithm(s) and rejects all other public key algorithms.

FCS_SSHS_EXT.1.5

The SSH server shall ensure that the SSH transport implementation uses [*hmac-sha2-256, hmac-sha2-512*] and [*no other MAC algorithms*] as its MAC algorithm(s) and rejects all other MAC algorithm(s).

FCS_SSHS_EXT.1.6

The SSH server shall ensure that [*diffie-hellman-group14-sha1, ecdh-sha2-nistp256*] and [*ecdh-sha2-nistp384, ecdh-sha2-nistp521*] are the only allowed key exchange methods used for the SSH protocol.

FCS_SSHS_EXT.1.7

The SSH server shall ensure that the SSH connection be rekeyed after [*no more than 1 Gigabyte of data has been transmitted, no more than 1 hour*] using that key.

5.2.2.14 FCS_TLSC_EXT.1 TLS Client Protocol

FCS_TLSC_EXT.1.1

The OS shall implement TLS 1.2 (RFC 5246) supporting the following cipher suites: [

- *TLS_RSA_WITH_AES_128_CBC_SHA as defined in RFC 5246,*
- *TLS_RSA_WITH_AES_256_CBC_SHA as defined in RFC 5246,*
- *TLS_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5246,*
- *TLS_RSA_WITH_AES_256_CBC_SHA256 as defined in RFC 5246,*
- *TLS_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5288,*
- *TLS_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5288,*
- *TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5246,*
- *TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 as defined in RFC 5246,*
- *TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5288,*
- *TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5288,*
- *TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5289,*
- *TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289,*
- *TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289,*
- *TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289,*
- *TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5289,*
- *TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289,*
- *TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289,*
- *TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289*

].

FCS_TLSC_EXT.1.2

The OS shall verify that the presented identifier matches the reference identifier according to RFC 6125.

FCS_TLSC_EXT.1.3

The OS shall only establish a trusted channel if the peer certificate is valid.

5.2.2.15 FCS_TLSC_EXT.2 TLS Client Protocol

FCS_TLSC_EXT.2.1 The OS shall present the Supported Groups Extension in the Client Hello with the following supported groups: [*secp256r1, secp384r1, secp521r1*].

5.2.3 User Data Protection (FDP)

5.2.3.1 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.

5.2.4 Identification and Authentication (FIA)

5.2.4.1 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*]**.

5.2.4.2 FIA_UAU.5 Multiple Authentication Mechanisms (Refined)

FIA_UAU.5.1

The **OS** shall provide the following authentication mechanisms [

- ***authentication based on user name and password,***
- ***for use in SSH only, SSH public key-based authentication as specified by the EP for Secure Shell***

] 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*

].

5.2.4.3 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 caSigning purpose in the key usage field
- The OS shall validate the revocation status of the certificate using [CRL as specified in RFC 5759]
- 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.

5.2.4.4 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 and [*HTTPS*] connections.

5.2.5 Security Management (FMT)

5.2.5.1 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.

5.2.5.2 FMT_SMF_EXT.1 Specification of Management Functions

FMT_SMF_EXT.1.1

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

Management Function	Administrator	User
Enable/disable [*screen lock*]	X	-
Configure [*screen lock*] inactivity timeout	X	-
Configure local audit storage capacity	X	-
Configure minimum password length	X	-
Configure minimum number of special characters in password	X	-
Configure minimum number of numeric characters in password	X	-
Configure minimum number of uppercase characters in password	X	-
Configure minimum number of lowercase characters in password	X	-
Configure lockout policy for unsuccessful authentication attempts through [*timeouts between attempts*]	X	-
Configure host-based firewall	X	-
Configure name/address of directory server with which to bind	-	-
Configure name/address of remote management server from which to receive management settings	-	-
Configure name/address of audit/logging server to which to send audit/logging records	X	-
Configure audit rules	X	-
Configure name/address of network time server	X	-
Enable/disable automatic software update	X	-
Configure WiFi interface	-	-
Enable/disable Bluetooth interface	-	-
Enable/disable [*no other external interfaces*]	-	-
[*no other management functions*]	-	-

Table 13 Management Functions

Application Note: "X" indicates the TOE implements the management function at the indicated privilege level.

5.2.6 Protection of the TSF (FPT)

5.2.6.1 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*]

5.2.6.2 FPT_ASLR_EXT.1 Address Space Layout Randomization

FPT_ASLR_EXT.1.1

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

5.2.6.3 FPT_SBOP_EXT.1 Stack Buffer Overflow Protection

FPT_SBOP_EXT.1.1

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

5.2.6.4 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,*

] .

5.2.6.5 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*

] .

5.2.6.6 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(3) 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(3).

5.2.6.7 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(3) 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(3) prior to installation.

5.2.7 TOE Access (FTA)

5.2.7.1 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.

5.2.8 Trusted path/channels (FTP)

5.2.8.1 FTP_ITC_EXT.1 Trusted channel communication

FTP_ITC_EXT.1.1

The OS shall use [

- *TLS as conforming to FCS_TLSC_EXT.1,*
- *SSH as conforming to the EP for Secure Shell*

] to provide a trusted communication channel between itself and authorized IT entities supporting the following capabilities: [*[application initiated TLS, 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.

5.2.8.2 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 and 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*]].

5.3 TOE SFR Dependencies Rationale for SFRs

[GPOSPP] and [SSHEP] contain all the requirements claimed in this Security Target. As such, the dependencies are not applicable since the PP, EP, and Module have been approved.

5.4 Security Assurance Requirements

The TOE assurance requirements for this ST are taken directly from [GPOSPP] which are derived from Common Criteria Version 3.1, Revision 5. The assurance requirements are summarized in the table below.

Assurance Class	Components	Components Description
Development	ADV_FSP.1	Basic Functional Specification
Guidance Documents	AGD_OPE.1	Operational User Guidance
	AGD_PRE.1	Preparative Procedures
Life Cycle Support	ALC_CMC.1	Labeling of the TOE
	ALC_CMS.1	TOE CM Coverage
	ALC_TSU_EXT.1	Timely Security Updates
Security Target evaluation	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_TSS.1	TOE summary specification
Tests	ATE_IND.1	Independent Testing – Conformance
Vulnerability Assessment	AVA_VAN.1	Vulnerability Analysis

Table 14 Security Assurance Requirements

5.5 Rationale for Security Requirements

A mapping of the Security Functional Requirements taken from the [GPOSPP] to the Security Objectives for the TOE can be found in Section 4.1 of the [GPOSPP]. This section also provides rationale for how SFR satisfies the objective.

FCS_SSH_EXT.1, FCS_SSHC_EXT.1, and FCS_SSHS_EXT.1 address O.PROTECTED_COMMS. The SFRs define the ability of the TOE to use the SSH protocol as a method of enforcing protected communications.

FCS_COP.1(1)/SSH addresses O.PROTECTED_COMMS. This SFR defines the cryptographic operation used to protect communications.

The Security Assurance Requirements were chosen because they are required by the [GPOSPP], and the ST claims exact conformance to the [GPOSPP].

5.6 Assurance Measures

The TOE satisfies the identified assurance requirements. This section identifies the Assurance Measures applied by Red Hat to satisfy the assurance requirements. The table below lists the details.

SAR Component	How the SAR will be met
ADV_FSP.1	The functional specification describes the external interfaces of the TOE; such as the means for a user to invoke a service and the corresponding response of those services. The description includes the interface(s) that enforces a security functional requirement, the interface(s) that supports the enforcement of a security functional requirement, and the interface(s) that does not enforce any security functional requirements. The interfaces are described in terms of their purpose (general goal of the interface), method of use (how the interface is to be used),

SAR Component	How the SAR will be met
	parameters (explicit inputs to and outputs from an interface that control the behavior of that interface), parameter descriptions (tells what the parameter is in some meaningful way), and error messages (identifies the condition that generated it, what the message is, and the meaning of any error codes).
AGD_OPE.1	The Administrative Guide provides the descriptions of the processes and procedures of how the administrative users of the TOE can securely administer the TOE using the interfaces that provide the features and functions detailed in the guidance.
AGD_PRE.1	The Installation Guide describes the installation, generation, and startup procedures so that the users of the TOE can put the components of the TOE in the evaluated configuration.
ALC_CMC.1 ALC_CMS.1	The Configuration Management (CM) documents describe how the consumer identifies the evaluated TOE. The CM documents identify the configuration items, how those configuration items are uniquely identified, and the adequacy of the procedures that are used to control and track changes that are made to the TOE. This includes details on what changes are tracked and how potential changes are incorporated.
ALC_TSU_EXT.1	Red Hat accepts reports of security issues at the secalert@redhat.com email address. Red Hat provides a public GPG key, 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. 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. 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.
ATE_IND.1	Red Hat will provide the TOE for testing.
AVA_VAN.1	Red Hat will provide the TOE for testing.

Table 15 TOE Security Assurance Measures

6 TOE Summary Specification

This chapter identifies and describes how the Security Functional Requirements identified above are met by the TOE.

TOE SFR	Rationale

FAU_GEN.1	The TOE generates and stores audit events using the Lightweight Audit Framework (LAF). The LAF is designed to be an audit system making the TOE compliant with the requirements from Common Criteria by intercepting all system calls and retrieving audit log entries from privileged user space applications. The framework allows configuring the events to be recorded from the set of all events that are possible to be audited and forwards the events matching the filters to the audit daemon. Each audit record contains the date and time of event, type of event, subject identity, and the user identity if applicable.
	
	Access to audit data by normal users is prohibited by the discretionary access control function of the TOE, which is used to restrict the access to the audit trail and audit configuration files to the system administrator only.
	
	An audit record consists of one or more lines of text containing fields in a "keyword=value" tagged format. The following information is contained in all audit record lines:
	
	• Type: indicates the source of the event, such as SYSCALL, PATH, USER_LOGIN, or LOGIN
	• Timestamp: Date and time (accurate to the millisecond) that the audit record was generated
	• Serial number: unique numerical event identifier appended to the timestamp
	• Login ID ("auid"), the user ID of the user authenticated by the system (regardless if the user has changed his real and / or effective user ID afterwards)
	• Effective user and group ID: the effective user and group ID of the process at the time the audit event was generated
	• Success or failure (where appropriate)
	• Process ID of the subject that caused the event (PID)
	• Hostname or terminal the subject used for performing the operation
	• Information about the intended operation
	• The success or failure of the action
	
	This information is followed by event specific data. In some cases, such as SYSCALL event records involving file system objects, multiple text lines will be generated for a single event, these all have the same time stamp and serial number to permit easy correlation.
	
	The TOE is able to generate audit records for the following events:
	
	• Start-up and shut-down of the audit function
	• Authentication Events
	• Use of privileged/special rights events:
	  ○ Security, audit, and configuration changes
	  ○ Privilege or role escalation
	  ○ Administrator or root-level access events
	  ○ User and Group management
	• File and object events (create, access, delete, modify, modify permissions)
	• Audit and log data access events
	• Cryptographic verification of software
	• System reboot, restart, and shutdown

TOE SFR	Rationale
	• Kernel module loading and unloading
FCS_CKM.1	The TOE implements RSA and ECC key generation as specified in FIPS 186-4. The TOE implements FFC key generation as specified in FIPS 186-4 and RFC 3526. RSA key sizes of 2048, 3072, and 4096 are supported. ECC curves P-256, P-384, and P-521 are supported. The FFC key size of L=2048, N=2047 (Group 14) is supported. For more detail, please see Table 5.
FCS_CKM.2	For RSA Key Establishment, the TOE implements Sections 7.1 and 7.2.1 of SP 800-56B. The TOE is a TLS client (i.e. sender), so it does not perform RSA decryption. The TOE supports 2048, 3072, and 4096-bit RSA keys. 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-256, P-384, and P-521 curves. For Finite field key establishment, the TOE implements Section 6.1.2.1 of SP 800-56A Rev. 3. The TOE supports finite field key establishment using group 14.
FCS_CKM_EXT.4	For volatile memory, the TOE destroys keys and key material by performing a single overwrite consisting of zeroes. For non-volatile memory, the TOE destroys keys and key material by performing an administrator configurable number (default 3) overwrites of the logical storage location with a pseudo random pattern. The pseudo random pattern is generated by an ISAAC PRNG which is initialized from /dev/urandom. See Section 6.1 for additional details.
FCS_COP.1(1) FCS_COP.1(1)/SSH	The TOE implements AES as specified in FIPS 197 with 128-bit and 256-bit key sizes. The TOE implements the following modes: CTR, CBC, GCM. For more detail, please see Table 5. The CTR mode counter is a 128-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. SSH rekeys at least every 512 MB of data transmitted using a key, so only a maximum of 2^25 counter values could be used, ensuring the counter does not wrap.
FCS_COP.1(2)	The TOE implements SHA-1, SHA-256, SHA-384, and SHA-512 as specified in FIPS 180-4. For more detail, please see Table 5.
FCS_COP.1(3)	The TOE implements RSA and ECDSA signature generation and verification as specified in FIPS 186-4. RSA key sizes of 2048, 3072, and 4096 are supported with SHA-1, SHA-256, SHA-384, and SHA-512. ECDSA curves P-256, P-384, and P-521 are supported with SHA-256, SHA-384, and SHA-512. For more detail, please see Table 5.
FCS_COP.1(4)	The TOE implements HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 as specified in FIPS 198-1. For more detail, please see Table 5.
FCS_RBG_EXT.1	The TOE generates random bits using a CTR_DRBG as specified in NIST SP 800-90A. For more detail, please see Table 5.
FCS_STO_EXT.1	The TOE includes the OpenSSL library to securely store sensitive data. OpenSSL provides file encryption services using AES-128 or AES-256 in CBC mode. Sensitive data include passwords and keys (Table 17) and can be found in /etc directory. /etc contains system-wide configuration files and system databases. Access to the files in /etc is limited with strict file permissions and/or encryption.

TOE SFR	Rationale

FCS_SSH_EXT.1
FCS_SSHC_EXT.1
FCS_SSHS_EXT.1
	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:
	  ○ aes128-ctr
	  ○ aes256-ctr
	  ○ aes128-cbc
	  ○ aes256-cbc
	• Public Key Algorithms:
	  ○ ssh-rsa
	  ○ rsa-sha2-256
	  ○ rsa-sha2-512
	  ○ ecdsa-sha2-nistp256
	  ○ ecdsa-sha2-nistp384
	• MACs:
	  ○ hmac-sha2-256
	  ○ hmac-sha2-512
	• Key Exchange Methods:
	  ○ diffie-hellman-group14-sha1
	  ○ ecdh-sha2-nistp256
	  ○ ecdh-sha2-nistp384
	  ○ ecdh-sha2-nistp521
	The TOE drops any SSH packet with a packet_length field greater than 262,144 bytes. The can TOE also rekey SSH connections before a key has been used for over an hour or used to protect more than 512 MB of data.
	OpenSSH utilizes algorithms provided by OpenSSL.

FCS_TLSC_EXT.1 FCS_TLSC_EXT.2 FCS_TLSC_EXT.4	The TOE provides three TLSv1.2 client implementations (OpenSSL and NSS). Each implementation supports the following ciphersuites: • TLS_RSA_WITH_AES_128_CBC_SHA as defined in RFC 5246, • TLS_RSA_WITH_AES_256_CBC_SHA as defined in RFC 5246, • TLS_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5246, • TLS_RSA_WITH_AES_256_CBC_SHA256 as defined in RFC 5246, • TLS_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5288, • TLS_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5288, • TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5246, • TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 as defined in RFC 5246, • TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5288, • TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5288, • TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5289, • TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289, • TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289, • TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289, • TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5289, • TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289, • TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289,

TOE SFR	Rationale
	• TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289 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 address, 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. The TOE presents the supported Elliptic Curves Extension in the Client Hello message with the P-256, P-384, and P-521 curves.
FDP_ACF_EXT.1	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 SAVETXT attribute is used for world-writable temp directories preventing the removal of files by users other than the owner. 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. The TOE also provides support for POSIX type ACLs to define a fine-grained access control on per-file or per-directory basis. An ACL entry contains the following information: • A tag type that specifies the type of the ACL entry • A qualifier that specifies an instance of an ACL entry type • A permission set that specifies the discretionary access rights for processes identified by the tag type and qualifier 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. 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. In addition, the following additional access control bits are processed by the kernel: • 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.

TOE SFR	Rationale

	• 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 is ignored.
	• SAVETXT: When a directory is marked with the SAVETXT 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 than the owning UID of the file system object.
	
	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/
	• 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/
	
	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.

FIA_AFL.1	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.
FIA_UAU.5	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. The TOE performs username and password authentication using a local set of credentials. Local username/password credentials are stored in the /etc/passwd and /etc/shadow files. The /etc/passwd file contains usernames, associated IDs, an indicator whether the password of the user is valid, the principal group id of the user and other (not security relevant) information. The /etc/shadow file contains a hash of the user's password, the user ID, the time the password was last changed, the expiration time, and the validity period of the password. Users are also warned to change their passwords at login time if the password will expire soon and are prevented from logging in if the password has expired. The time of the last successful logins is recorded in the directory /var/log/faillock where one file per user is kept. Users can change their own password. Only administrators can

TOE SFR	Rationale
	add or delete users or change their properties. OpenSSH server is able to perform a 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.
FIA_X509_EXT.1 FIA_X509_EXT.2	The TOE uses X.509v3 certificates as defined by RFC 5280 to support authentication for TLS and HTTPS connections. The X.509 certificates are validated using the certificate path validation algorithm defined in RFC 5280, which can be summarized as follows: • the public key algorithm and parameters are checked • the current date/time is checked against the validity period • revocation status is checked using CRL • issuer name of X matches the subject name of X+1 • extensions are processed The certificate validity check is performed when the TOE receives the certificate during a TLS handshake. When the certificate being validated is for a TLS server, the TOE ensures the Extended Key Usage extension contains the Server Authentication purpose. When the certificate being validated is for an OCSP response, the TOE ensures the Extended Key Usage extension contains the OCSP signing purpose. The TOE ensures all CA certs contain the basic constraints extension and that the CA=TRUE flag is set. 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).
FMT_MOF_EXT.1	The TOE restricts all "Administrator" management activities listed in FMT_SMF_EXT.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.
FMT_SMF_EXT.1	The TOE allows the administrators to perform the following management activities: • Enable/disable screen lock • Configure screen lock inactivity timeout • Configure local audit storage capacity • Configure minimum password length • Configure minimum number of special characters in password • Configure minimum number of numeric characters in password • Configure minimum number of uppercase characters in password • Configure minimum number of lowercase characters in password • Configure lockout policy for unsuccessful authentication attempts through timeouts between attempts • Configure host-based firewall • Configure name/address of directory server with which to bind • Configure name/address of audit/logging server to which to send audit/logging records

TOE SFR	Rationale
	• Configure audit rules • Configure name/address of network time server • Enable/disable automatic software update Non-administrative users are not allowed to manage the TOE.
FPT_ACF_EXT.1	The TOE uses the file/directory permissions described in FDP_ACF_EXT.1 to prevent unprivileged users from modifying: • Kernel and its drivers/modules • Security audit logs • Shared libraries • System executables • System configuration files
FPT_ASLR_EXT.1	The TOE executables are compiled as Position Independent Executables with the following amount of randomization: • exec 30 bits • heap 30 bits • so 29 bits • mmap 29 bits • stack 30 bits The TOE developer guidance instructs developers to compile non-TOE executables with PIE flags, so non-TOE executables have the same amount of randomization.
FPT_SBOP_EXT.1	The TOE is compiled with the option “stack-protector-strong” to add a stack canary and associated verification code during the entry and exit of function frames to prevent stack-based buffer overflows.
FPT_SRP_EXT.1	The TOE restricts execution of programs to those allowed by the configured Application Whitelisting rules. The Application Whitelisting rules are configured to restrict execution based on file path. The file paths of whitelisted applications are added to the trust database as part of the installation process for each application.
FPT_TST_EXT.1	The Unified Extensible Firmware Interface (UEFI) Secure Boot technology ensures that the system firmware checks whether the system boot loader is signed with a cryptographic key authorized by a database of public keys contained in the firmware. With signature verification in the next-stage boot loader and kernel, it is possible to prevent the execution of kernel space code which has not been signed by a trusted key. The signature on the first-stage boot loader (shim.efi) is verified to be signed by a certificate authority (CA) stored in the firmware database. shim.efi then uses an embedded RSA 2048 public key to verify the signature on the RSA 2048 code signing public key. This code signing key is used to verify the signature of the second-stage boot loader, GRUB 2 (grubx64.efi). Finally, GRUB 2 uses the code signing key to verify the signature on the OS kernel before passing control to the kernel. The kernel has 2 more embedded keys that are used to authenticate drivers and kernel modules.
FPT_TUD_EXT.1 FPT_TUD_EXT.2	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.
FTA_TAB.1	The TOE can be configured to display an administrator configured advisory warning message prior to establishing a local or remote interactive user session.

TOE SFR	Rationale
FTP_ITC_EXT.1	The TOE provides a TLS Client protocol implementation which allows applications to protect communications with remote IT entities. The TOE uses the SSH server protocol to protect the communications with remote users. The TOE also allows users to securely connect to remote servers using SSH.
FTP_TRP.1	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.

Table 16 TOE Summary Specification SFR Description

6.1 Cryptographic Keys

Key	Type	Volatile Management	Non-Volatile Storage
TLS Diffie-Hellman Private Key	FFC Group 14 Or ECC P-256, P-384, or P-521	Generated by the DRBG as specified by FCS_CKM.1 and FCS_CKM.2	N/A
TLS Pre-Master Secret	Data used to derive keys	Generated by the DRBG Or Established using Diffie-Hellman (FFC & ECC)	N/A
TLS Session Keys	AES 128-bit or 256-bit And HMAC 160-bit, 256-bit, or 384-bit	Derived from the TLS Pre-Master Secret	N/A
SSH Server Private Key	RSA 2048, 3072, or 4096 Or ECDSA P-256 or P-384	Loaded from the filesystem	Storage method: Filesystem API
SSH User Private Key	RSA 2048, 3072, or 4096 Or ECDSA P-256 or P-384	Loaded from the filesystem	Storage method: Filesystem API
SSH Diffie-Hellman Private Key	FFC Group 14 Or ECC P-256, P-384, or P-521	Generated by the DRBG as specified by FCS_CKM.1 and FCS_CKM.2	N/A
SSH Shared Secret	Data used to derive keys	Established using Diffie-Hellman (FFC & ECC)	N/A
SSH Session Keys	AES 128-bit or 256-bit And HMAC 256-bit or 512-bit	Derived from the SSH Shared Secret	N/A
User Passwords	ASCII text	Entered by the user	N/A – Salted and hashed passwords are stored in /etc/shadow
File Encryption Key	AES 128-bit or 256-bit	Loaded from the filesystem Or Entered by the user Or Derived from a password entered by the user	N/A

Table 17 Cryptographic Keys

6.2 Stack Smashing Protection

The TOE includes a number of 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.

The following items are data tables for character set conversion in glibc. They do not require stack smashing protection:

/usr/lib64/gconv/libCNS.so

/usr/lib64/gconv/libGB.so

/usr/lib64/gconv/libISOIR165.so

/usr/lib64/gconv/libJIS.so

/usr/lib64/gconv/libJISX0213.so

/usr/lib64/gconv/libKSC.so

The following items are kernel modules that have handwritten assembler. They do not require stack smashing protection:

/usr/lib/modules/4.18.0-147.el8.x86_64/vdso/vdso32.so

/usr/lib/modules/4.18.0-147.el8.x86_64/vdso/vdso64.so

The following item is the run time linker which has special needs/handwritten assembler:

/usr/lib64/ld-2.28.so

The following item is a vector math library. It has a few short functions and does math operations on arguments being passed on. There are no variables that need stack smashing protection:

/usr/lib64/libmvec-2.28.so