Red Hat, Inc. Red Hat Enterprise Linux Security Target

Document Version: 1.1

2400 Research Blvd

Suite 395

Rockville, MD 20850

Revision History

Version	Date	Changes
Version 0.1	June 2022	Initial Release
Version 0.1.1	June 2022	Updated based on PKG_SSH inclusion
Version 0.2	June 2022	Update based on feedback
Version 0.3	June 2022	Update based on vendor feedback
Version 0.4	August 2022	Updated based on initial validator feedback
Version 0.5	August 2022	Updates to ASLR and Stack Smashing TSS narratives
Version 0.5.1	September 2022	Typo Fixes
Version 0.6	March 2023	Updated based on vendor feedback; version change from 8.4 to 8.6
Version 0.7	April 2023	Document name change; update based on vendor feedback
Version 0.8	September 2023	Removed session timeout, session inactivity timeout and lockout policy.
Version 0.9	November 2023	Updated TDs
Version 1.0	November 2023	Final version for submission.
Version 1.1	January 2024	Addressed ECR comments.

Contents

List of Tables

List of Figures

No table of figures entries found.

1 Introduction

The Security Target (ST) serves as the basis for the Common Criteria (CC) evaluation and identifies the Target of Evaluation (TOE), the scope of the evaluation, and the assumptions made throughout. This document will also describe the intended operational environment of the TOE, and the functional and assurance requirements that the TOE meets.

1.1 Security Target and TOE Reference

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

Table 1 – TOE/ST Identification

Category	Identifier
ST Title	Red Hat Enterprise Linux 8.6 Security Target
ST Version	1.1
ST Date	January 2024
ST Author	Acumen Security, LLC
TOE Identifier	Red Hat Enterprise Linux Extended Update Support
TOE Version	8.6
TOE Developer	Red Hat, Inc.
Key Words	Operating System, SSH, TLS, Linux

1.2 TOE Overview

Red Hat® Enterprise Linux® is an open-source operating system (OS) that supports multiple users, user permissions, access controls, and cryptographic functionality.

1.3 TOE Description

This section provides an overview of the TOE, including physical boundaries, security functions, and relevant TOE documentation and references.

1.3.1 Physical Boundaries

The TOE itself is an operating system which can be installed on any compatible hardware; as such, the TOE does not have physical boundaries. However, the TOE was evaluated on the following hardware:

Table 2 – Hardware Platforms

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 R840	Xeon Silver 42xx
Dell Inc.	PowerEdge R940	Xeon Silver 42xx

Vendor	Model	CPU
Dell Inc.	PowerEdge R940xa	Xeon Silver 42xx
IBM	z15 8561-T01	IBM z15
IBM	z15 8562-T02	IBM z15
IBM	z15 8561-LT1	IBM z15
IBM	z15 8562-LT2	IBM z15

Dell Platforms:

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.

IBM Platforms:

The TOE is one instance of RHEL 8 running on an abstract machine and has full control over the abstract machine inside an IBM z15 T01, T02, LT1, or LT2 mainframe (machine type 8561 or 8652). The abstract machine is provided by a logical partition of the z15 processor. The partition includes 5 IFL (Integrated Facility for Linux) processors. An IFL is a processor dedicated to and optimized for Linux workloads. Because of SMT, the IFL's appear as 10 logical processors allocated to the partition.

The TOE was tested on a IBM z15 T01 mainframe machine type 8561.

1.3.2 Security Functions Provided by the TOE

The TOE provides the security functions required by PP_OS_V4.2.1 and PKG_SSH_V1.0.

1.3.2.1 Security Audit

The TOE generates and stores audit events locally using administrator defined rules.

1.3.2.2 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, and in full detail in Section 6.2 of this document.

Table 3 TOE Cryptographic Protocols

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.

The TOE includes the OpenSSL cryptographic library, and each cryptographic algorithm has been validated for conformance to the requirements specified in their respective standards as identified in Section 6.2 of this document.

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.

The TOE also provides a kernel cryptographic API (KCAPI), which implements an SP 800-90A compliant HMAC_DRBG to generate high-security random output for key generation or seed material.

1.3.2.3 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.

1.3.2.4 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.

1.3.2.5 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.

1.3.2.6 TOE Access

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

1.3.2.7 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):

- Address Space Layout Randomization for user space code.
- Kernel and user-space ring-based separation of processes
- Stack buffer overflow protection using stack canaries.
- Secure Boot ensures 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.

1.3.2.8 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.

1.3.3 TOE Documentation

The following documents are essential to understanding and controlling the TOE in the evaluated configuration:

- [ST] Red Hat Enterprise Linux 8.6 Security Target, Version 1.0
- [AGD] Red Hat Enterprise Linux 8.6 CC Guidance, Version 3.4

1.3.4 References

In additional to TOE documentation, the following reference may also be valuable when understanding and controlling the TOE:

- Protection Profile for General Purpose Operating Systems, Version 4.2.1 [PP_OS_V4.2.1]
- Functional Package for SSH, Version 1.0 [PKG_SSH_V1.0]

1.4 TOE Environment

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

Table 4 - Operational Environment Components

Component	Required	Usage/Purpose/Description for TOE Performance
Workstation with SSH Client	Yes	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.
Update Server	Yes	Provides the ability to check for updates to the TOE as well as providing signed updates.

1.5 Product Functionality not Included in the Scope of the Evaluation

The following product functionality is not included in the CC evaluation:

- SELinux Mandatory Access Control System
- OS Virtualization Infrastructure
- Containerization infrastructure

2 Conformance Claims

This section identifies the TOE conformance claims, conformance rationale, and relevant Technical Decisions (TDs).

2.1 CC Conformance Claims

The TOE is conformant to the following:

- 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 extended
- Common Criteria for Information Technology Security Evaluations Part 3, Version 3.1, Revision5, April 2017 extended

2.2 Protection Profile Conformance

This ST claims exact conformance to:

- Protection Profile for General Purpose Operating Systems, Version 4.2.1, April 22, 2019
- Functional Package for Secure Shell (SSH), Version 1.0, May 13, 2021

2.3 Conformance Rationale

The security problem definition, security objectives, and security requirements in this ST are all taken from the PP and Functional Package, performing only the operations defined there.

2.3.1 Technical Decisions

All NIAP TDs issued to date and applicable to the PP and Functional Package have been considered. Table 5 identifies all applicable TDs.

Table 5 – Relevant Technical Decisions

Technical Decision	Applicable (Y/N)	Exclusion Rationale (if applicable)
Technical Decisions applicable to General Purpose Operating Systems Protection Profile v 4.2.1		
0715 – Updates to FIA_X509_EXT.1 for Exception Processing and Test Conditions	Yes	
0680 - OS 4.2.1 Conformance Claims section updated to allow for MOD_WLAN_CLI_v1.0	No	Evaluation does not include MOD_WLAN_CLI_v1.0.
0649 – Conformance Claims for OS PP v4.2.1	Yes	
0630 – FCS_COP.1 requirements for Secure Shell	Yes	
0600 – Conformance claim sections updated to all for MOD_VPNC_V2.3	No	Evaluation does not include MOD_VPNC_V2.3.
0578 – SHA-1 is no longer mandatory	Yes	
0501 – Cryptographic selections and updates for OS PP	Yes	

Technical Decision	Applicable (Y/N)	Exclusion Rationale (if applicable)
0493 – X.509v3 certificates when using digital signatures for Boot Integrity	Yes	
0463 – Clarification for FPT_TUD_EXT	Yes	
0441 – Updated TLS Ciphersuites for OS PP	Yes	
0386 – Platform-Provided Verification of Update	Yes	
0365 – FCS_CKM_EXT.4 selections	Yes	
Technical Decisions applicable to Secure Shell (SSH) Functional Package v 1.0		
TD0777 – Clarification to Selections for Auditable Events for FCS_SSH_EXT.1	Yes	
TD0732 – FCS_SSHS_EXT.1.3 Test 2 Update	Yes	
TD0695 – Choice of 128 or 256 bit size in AES-CTR in SSH Functional Package.	Yes	
TD0682 – Addressing Ambiguity in FCS_SSHS_EXT.1 Tests	Yes	

3 Security Problem Definition

The security problem definition is taken directly from the claimed PP and Functional Package specified in Section 2.2 and 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 (OSPs) that the TOE is expected to enforce.

3.1 Threats

The threats included in Table 6 are drawn directly from the PP and Functional Package specified in Section 2.2.

Table 6 – Threats

ID	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.

3.2 Assumptions

The assumptions included in Table 7 are drawn directly from the PP and Functional Package.

Table 7 – Assumptions

ID	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.

3.3 Organizational Security Policies

The PP and Functional Package do not define any Organizational Security Policies (OSPs).

4 Security Objectives

The security objectives have been taken directly from the claimed PP and Functional Package and are reproduced here for the convenience of the reader.

4.1 Security Objectives for the TOE

The security objectives in the following table apply to the TOE:

Table 8 – Security Objectives for the TOE

ID	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.MANAGMENT	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

Security objectives for the operational environment assist the TOE in correctly providing its security functionality. These objectives, which are found in the table below, track with the assumptions about the TOE operational environment.

Table 9 – Security Objectives for the Operational Environment

ID	Description
OE.PLATFORM	The OS relies on being installed on trusted hardware.

ID	Description
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.

5 Security Requirements

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

Table 10 – Security Functional Requirements

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(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/Intel	TLS Client Protocol on Intel Platforms
FCS_TLSC_EXT.1/z15	TLS Client Protocol on z15 Platforms
FCS_TLSC_EXT.2	TLS Client Protocol on Intel Platforms
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

5.1 Conventions

The CC allows the following types of operations to be performed on the functional requirements: assignments, selections, refinements, and iterations. The following font conventions are used within this document to identify operations defined by CC:

- Assignment: Indicated with *italicized* text;
- Refinement: Indicated with **bold** text;
- Selection: Indicated with underlined text;
- Iteration: Indicated by appending the iteration identifier in parenthesis, e.g., (1), (2), (3).
- Where operations were completed in the PP and relevant EPs/Modules/Packages, the formatting used in the PP has been retained.
- Extended SFRs are identified by the addition of “EXT” after the requirement name.
- Where SFRs are categorized by the same SFR title with a separate names appended, the /NAME format is used.

5.2 Security Functional Requirements

This section includes the security functional requirements for this ST.

5.2.1 Audit Data Generation (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),*
     - *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 11].*

]

].

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 11].

Table 11 – SSH Auditable Events

Requirement	Auditable Events	Additional Audit Record Contents
FCS_SSH_EXT.1	**[**None**]**	**[**None**]**
FCS_SSH_EXT.1	**[**Establishment of SSH connection, None**]**	**[**Non-TOE endpoint of connection (IP Address)**]**
FCS_SSH_EXT.1	**[**Termination of SSH connection session, None**]**	**[**Non-TOE endpoint of connection (IP Address)**]**
FCS_SSH_EXT.1	**[**None**]**	**[**None**]**
FCS_SSHC_EXT.1	No events specified	

Requirement	Auditable Events	Additional Audit Record Contents
FCS_SSHS_EXT.1	No events specified	

Application Note: This SFR has been updated according to TD0777

5.2.2 Cryptographic Support (FCS)

5.2.2.1 FCS_CKM.1Cryptographic Key Generation (Refined)

FCS_CKM.1.1 (Refined)

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 Diffie-Hellman group 14 that meet the following: RFC 3526,*
- *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]~~ .

Application Note: This SFR has been updated according to TD0501.

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: [

- ***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",***
- ***Key establishment scheme using Diffie-Hellman group 14 that meets the following: RFC 3526***

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

Application Note: This SFR has been updated according to TD0501.

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: [*
  - *logically addresses the storage location of the key and performs a [[administrator specified number (default of 3) of]] overwrite consisting of [pseudo-random pattern],*

]

] .

Application Note: This SFR has been updated according to TD0365.

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)***
- ***AES-CTR (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].~~

Application Note: This SFR has been updated according to TD0630

5.2.2.5 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*].

Application Note: This SFR has been updated according to TD0578.

5.2.2.6 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].~~

5.2.2.7 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-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*].

5.2.2.8 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)*
- *HMAC_DRBG (SHA-512)*

].

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.

5.2.2.9 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.

5.2.2.10 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, 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-nistp256 (RFC 5656),*
  - *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: [

- *aes128-ctr (RFC 4344),*
- *aes256-ctr (RFC 4344),*
- *aes128-cbc (RFC 4253),*
- *aes256-cbc (RFC 4253),*
- *aes128-gcm@openssh.com (RFC 5647),*
- *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-group14-sha256,*
- *diffie-hellman-group16- sha512,*

- *diffie-hellman-group18-sha512*
- *ecdh-sha2-nistp256 (RFC 5656),*
- *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 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.

5.2.2.11 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-nistp256 (RFC 5656),*
  - *ecdsa-sha2-nistp384 (RFC 5656),*
  - *ecdsa-sha2-nistp521 (RFC 5656)*

],

] as described in RFC 4251 section 4.1.

5.2.2.12 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-nistp256 (RFC 5656),*
- *ecdsa-sha2-nistp384 (RFC 5656),*
- *ecdsa-sha2-nistp521 (RFC 5656)*

].

5.2.2.13 FCS_TLSC_EXT.1 TLS Client Protocol on Intel

FCS_TLSC_EXT.1.1/Intel

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

- *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_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_GCM_SHA384 as defined in RFC 5289*

].

Application Note: This SFR has been updated according to TD0441.

FCS_TLSC_EXT.1.2/Intel

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

FCS_TLSC_EXT.1.3/Intel

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

5.2.2.14 FCS_TLSC_EXT.1 TLS Client Protocol on z15

FCS_TLSC_EXT.1.1/z15

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

- *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,*

].

FCS_TLSC_EXT.1.2/z15

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

FCS_TLSC_EXT.1.3/z15

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 on Intel

FCS_TLSC_EXT.2.1/Intel

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

Application Note: This SFR only applies to Intel-based hardware platforms which select ECDHE ciphersuites.

5.2.2.16 FCS_TLSC_EXT.3 TLS Client Protocol on z15

FCS_TLSC_EXT.3.1/z15

The OS 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.

Application Note: This requirement limits the hashing algorithms supported for the purpose of digital signature verification by the client and limits the server to the supported hashes for the purpose of digital signature generation by the server. The

signature_algorithm extension is only supported by TLS 1.2.

5.2.2.17 FCS_TLSC_EXT.3 TLS Client Protocol on Intel

FCS_TLSC_EXT.3.1/Intel

The OS 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.

Application Note: This requirement limits the hashing algorithms supported for the purpose of digital signature verification by the client and limits the server to the supported hashes for the purpose of digital signature generation by the server. The

signature_algorithm extension is only supported by TLS 1.2.

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 Functional Package for Secure Shell***

] to support user authentication.

Application Note: This SFR has been updated according to TD0649.

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 8603] 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*].

Application Note: This SFR has been updated according to TD0604

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:

Table 12 – Specification of Management Functions

Management Function	Administrator	User
Enable/disable [*session timeout*]	-	-
Configure [*session*] inactivity timeout	-	-
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	-

Management Function	Administrator	User
Configure lockout policy for unsuccessful authentication attempts through [*timeouts between attempts*]	-	-
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*]	-	-

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 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*].

5.2.6.3 FPT_ASLR_EXT.1 Address Space Layout Randomization on IBM z15

FPT_ASLR_EXT.1.1/z15

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

5.2.6.4 FPT_SBOP_EXT.1 Stack Buffer Overflow Protection

FPT_SBOP_EXT.1.1

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

5.2.6.5 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,

].

5.2.6.6 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

] .

Application Note: This SFR has been updated according to TD0493.

5.2.6.7 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.

Application Note: This SFR has been updated according to TD0463.

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).

Application Note: This SFR has been updated according to TD0386.

5.2.6.8 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.

Application Note: This SFR has been updated according to TD0463.

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 TOA 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 Functional Package 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.

Application Note: This SFR has been updated according to TD0649.

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, 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 Security Assurance Requirements

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 13.

Table 13 – Security Assurance Requirements

Assurance Class	Assurance Components	Component Description
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
Development	ADV_FSP.1	Basic Functionality 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
Tests	ATE_IND.1	Independent Testing – Conformance
Vulnerability Assessment	AVA_VAN.1	Vulnerability Survey

5.4 Dependencies

PP_OS_V4.2.1 and PKG_SSH_V1.0 contain all the requirements claimed in this Security Target. As such, the dependencies are not applicable since the PP and Functional Package have been approved.

5.5 Security Objectives Rationale

A mapping of the Security Functional Requirements taken from PP_OS_V4.2.1 to the Security Objectives for the TOE can be found in Section 4.1 of PP_OS_V4.2.1. This section also provides rationale for how SFRs 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) 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 [PP_OS_V4.2.1], and the ST claims exact conformance to PP_OS_V4.2.1 and PKG_SSH_V1.0

6 TOE Summary Specification

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

Table 14 – TOE Summary Specification

Requirement	TSS Description
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. Security updates are delivered via the standard update mechanism described in FPT_TUD_EXT.1.
FAU_GEN.1	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. 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.

Requirement	TSS Description
	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: • Type: indicates the kind of the information in the record, such as SYSCALL, PATH, USER_LOGIN, or LOGIN • Timestamp: Date and time (accurate to the millisecond) that the audit record was generated • Serial number: a unique numerical identifier appended to the timestamp to separate events within the same millisecond. • 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. • Session ID: A unique identifier to disambiguate which login session an event belongs to. • Uid: the real user ID of the process at the time the audit event was generated • Pid: the process ID of the subject that caused the event. • Res: Success or failure results There can be optional information depending on the kind of the event which may include, but is not limited to: • The system call that a process made that caused the event • The group ID of the subject • Hostname or terminal the subject used for performing the operation • Information about the intended operation
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 ST section 6.2
FCS_CKM.2	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.

Requirement	TSS Description
FCS_CKM_EXT.4 *Updated according to TD0365*	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)	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 ST section 6.2 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. 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 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-256, SHA-384, and SHA-512 as specified in FIPS 180-4. For more detail, please see Table 15.
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-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 ST section 6.2
FCS_COP.1(4)	The TOE implements HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 as specified in FIPS 198-1. For more detail, please see ST section 6.2
FCS_RBG_EXT.1	The TOE uses two DRBGs: • SP 800-90A HMAC_DRBG(SHA-512) in the Kernel • SP 800-90A CTR_DRBG(AES-256) in OpenSSL The kernel HMAC_DRBG is seeded with at least 256-bits of entropy from the platform-provided noise sources. The OpenSSL CTR_DRBG is seeded with at least 256-bits of entropy from the kernel HMAC_DRBG.

Requirement	TSS Description
	This DRBG is used to generate session and ephemeral keys during SSH and TLS protocol negotiation, and for all other uses of entropy (such as the generation of nonces).
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 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.

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
	  ○ aes128- gcm@openssh.com
	  ○ aes256-gcm@openssh.com
	• Public Key Algorithms:
	  ○ rsa-sha2-256
	  ○ rsa-sha2-512
	  ○ ecdsa-sha2-nistp256
	  ○ ecdsa-sha2-nistp384
	  ○ ecdsa-sha2-nistp521
	• MACs:
	  ○ hmac-sha2-256
	  ○ hmac-sha2-512
	  ○ implicit
	• Key Exchange Methods:
	  ○ ecdh-sha2-nistp256
	  ○ ecdh-sha2-nistp384
	  ○ ecdh-sha2-nistp521
	• Host Key Algorithms:
	  ○ rsa-sha2-256
	  ○ rsa-sha2-512
	  ○ ecdsa-sha2-nistp256
	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

Requirement	TSS Description
	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 512 MB of data. OpenSSH utilizes algorithms provided by OpenSSL.
FCS_TLSC_EXT.1 /Intel FCS_TLSC_EXT.1/z15 FCS_TLSC_EXT.2	On Intel-based platforms, the TOE provides a TLSv1.2 client implementation with the following ciphersuites: • 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_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_GCM_SHA384 as defined in RFC 5289 On Intel-based platforms, the TOE presents the supported Elliptic Curves Extension in the Client Hello message with the P-256, P-384, and P-521 curves. On z15-based platforms, the TOE provides a TLSv1.2 client implementation with the following ciphersuites: • 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, On both platforms, 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

Requirement	TSS Description
	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.
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 a 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

Requirement	TSS Description

	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.
	• 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 from 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	None The TOE will detect when an administrator configurable integer within 1-65,535 unsuccessful authentication attempts for

Requirement	TSS Description
	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. 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. 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. The OpenSSH server is able to perform key-based authentication. When a user wants to log in, instead of providing a password, the ssh client application 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 *Updated according to TD0604* 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

Requirement	TSS Description
	• 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. 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: • 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 host-based firewall • Configure name/address of audit/logging server to which to send audit/logging records • 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

Requirement	TSS Description
	• System executables • System configuration files
FPT_ASLR_EXT.1/Xeon FPT_ASLR_EXT.1/z15	On Intel Xeon Silver processors, 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 On IBM z15 architecture, the TOE executables are compiled as Position Independent Executables with the following amount of randomization: • exec 11 bits • heap 19 bits • so 13 bits • mmap 13 bits • stack 21 bits On both 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.
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. The compiler guards functions that call “alloca”, or with buffers larger than or equal to 8 bytes, or those that have local array definitions, or have references to local frame addresses. Only variables that are actually allocated on the stack are considered. Optimized away variables or variables allocated in registers are not considered.
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	Dell: The Unified Extensible Firmware Interface (UEFI) Secure Boot technology verifies the signature on the system boot loader is signed

Requirement	TSS Description
	valid before executing the system boot loader. The signature is considered valid if the signing key is present in the 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. IBM: Linux Secure Boot on z15 is based on List Directed Initial Program Load (LD-IPL) processing. LD-IPL is used for booting Operating Systems from SCSI/FCP and NVMe devices. For Linux this includes a kernel image, an initial RAM disk, kernel parameters, and a boot loader. The system firmware checks whether a Secure Boot has been requested by checking a parameter in the IPL Parameter Block. If so, the machine loader will perform additional steps in order to validate the integrity of the image to be booted. The machine loader locates the signature data stored on disk. It then works through a list of certificates used to verify the signature, stopping on the first matching certificate. These certificates can only be changed through firmware updates of the machine loader. The loader locates the address of the program (in this case, the Stage 3 Bootloader) to be booted and verifies that its signature matches the certificate. If it does, then it executes the program. Other parts of the boot chain and the operating system can find the certificate by looking at absolute address 14 in the IPL Parameter Block. This can be used to locate the IPL Information Report Block where the certificate is located. The machine loader then turns control over to the TOE. The TOE controlled boot process begins with the Stage 3 Bootloader (a component of the larger zipl bootloader). The Stage 3 Bootloader reads the signature and RSA 4096-bit public key for the Kernel from the IPL Parameter Block. The Stage 3 Bootloader attempts to verify the signature of the Kernel. If the signature verification succeeds, the Stage 3 Bootloader passes control to the Kernel. If there is no

Requirement	TSS Description
	signature, or signature verification failed, the Stage 3 Bootloader terminates the boot.
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. Updates to the TOE and application software are downloaded by the TOE from the Red Hat CDN.
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.
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.

6.1 Cryptographic Keys

Table 15 - Cryptographic Key Details

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	Established using Diffie-Hellman (FFC & ECC)	N/A
TLS Session Keys	AES 128-bit or 256-bit And HMAC 256-bit or 384-bit	Derived from the TLS Pre-Master Secret	N/A
SSH Server Private Key	RSA 2048, 3072, or 4096	Loaded from the filesystem	Storage method: Filesystem API

	Or ECDSA P-256 or P-384		
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	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 (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

6.2 CAVP Algorithm Certificate Details

Each of these cryptographic algorithms have been validated as identified in the table below:

Table 16 - CAVP Algorithm Testing References for the TOE

Algorithm	Related SFRs	Implementation	TOE Use	CAVP Certificate #
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	OpenSSL	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	A1794, A1794, A2781, A1816 A1794
Diffie-Hellman	FCS_CKM.1 FCS_CKM.2 FCS_TLSC_EXT.1	OpenSSL	TLS Diffie-Hellman Group 14 Key Establishment	No NIST CAVP.
DRBG	FCS_RBG_EXT.1	OpenSSL	CTR_DRBG (AES-256)	A1794
		Red Hat Enterprise Linux 8 Kernel Crypto API Cryptographic Module	HMAC-DRBG (SHA-512)	A4710
ECDSA	FCS_CKM.1 FCS_COP.1(3) FCS_SSHC_EXT.1 FCS_SSHS_EXT.1 FCS_TLSC_EXT.1 FCS_TLSC_EXT.2	OpenSSL	SSH ECDSA P-256 and P-384 Host Key and User 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 ECDSA P-256, P-384, and P-521 Signature Generation and Verification	A1823 A1823 A1823 A1823
HMAC	FCS_COP.1(4) FCS_SSHC_EXT.1 FCS_SSHS_EXT.1 FCS_TLSC_EXT.1	OpenSSL	SSH HMAC-SHA-256 and HMAC-SHA-512 TLS HMAC-SHA-256, and HMAC-SHA-384	A1823 A1823

Algorithm	Related SFRs	Implementation	TOE Use	CAVP Certificate #
	FCS_COP.1(4)	Red Hat Enterprise Linux 8 Kernel Crypto API Cryptographic Module	HMAC-SHA-512	A4710
KAS-FFC-SSC	FCS_CKM.2 FCS_TLSC_EXT.2	OpenSSL	TLS FFC Diffie-Hellman Key Establishment	A1834
KAS-ECC-SSC	FCS_CKM.2 FCS_SSHC_EXT.1 FCS_SSHS_EXT.1 FCS_TLSC_EXT.2	OpenSSL	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	A1823 A1823
RSA	FCS_CKM.1 FCS_CKM.2 FCS_COP.1(3) FCS_SSHC_EXT.1 FCS_SSHS_EXT.1 FCS_TLSC_EXT.1 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 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	A1823 A1823 A1823 A1823 A1823 A1823 A1823
SHS	FCS_COP.1(2)	Red Hat Enterprise Linux 8 Kernel Crypto API Cryptographic Module	SHA-256, SHA-384, and SHA-512 Hash	A4710
	FCS_COP.1(2) FCS_SSHC_EXT.1 FCS_SSHS_EXT.1	OpenSSL	SHA-256, SHA-384, and SHA-512 Hash SHA-256, SHA-384, and SHA-512 for Digital Signatures and HMACs	A1823 A1823

6.3 Stack Smashing Protection

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.

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

- /usr/lib64/ld-2.28.so
- /usr/lib64/libc-2.28.so
- /usr/lib64/Mcrt1.o
- /usr/lib64/Scrt1.o
- /usr/lib64/crt1.o
- /usr/lib64/gcrt1.o
- /usr/sbin/build-locale-archive
- /usr/sbin/ldconfig
- /usr/lib64/libpthread-2.28.so
- /usr/lib64/ld-2.28.so
- /usr/lib64/libc-2.28.so
- /usr/lib64/libm-2.28.so
- /usr/lib64/libmvec-2.28.so
- /usr/lib64/librt-2.28.so
- /usr/lib64/Scrt1.o
- /usr/lib64/crt1.o
- /usr/lib64/gcrt1.o
- /usr/lib64/grcrt1.o
- /usr/lib64/rcrt1.o
- /usr/lib64/Mcrt1.o

The following glibc files setup caches of information used application startup, so they are not invoked during normal application operations.

- /usr/sbin/build-locale-archive
- /usr/sbin/ldconfig

gcc has special handwritten assembler that establishes the C runtime environment before a program's main() function is invoked. As such, it cannot enable the compiler-based stack protector.

- /usr/lib/gcc/x86_64-redhat-linux/8/32/crtbegin.o
- /usr/lib/gcc/x86_64-redhat-linux/8/32/crtbeginS.o
- /usr/lib/gcc/x86_64-redhat-linux/8/32/crtbeginT.o
- /usr/lib/gcc/x86_64-redhat-linux/8/32/crtend.o
- /usr/lib/gcc/x86_64-redhat-linux/8/32/crtendS.o
- /usr/lib/gcc/x86_64-redhat-linux/8/32/crtfastmath.o
- /usr/lib/gcc/x86_64-redhat-linux/8/32/crtoffloadbegin.o
- /usr/lib/gcc/x86_64-redhat-linux/8/32/crtoffloadend.o
- /usr/lib/gcc/x86_64-redhat-linux/8/32/crtoffloadtable.o
- /usr/lib/gcc/x86_64-redhat-linux/8/32/crtprec32.o
- /usr/lib/gcc/x86_64-redhat-linux/8/32/crtprec64.o

- /usr/lib/gcc/x86_64-redhat-linux/8/32/crtprec80.o
- /usr/lib/gcc/x86_64-redhat-linux/8/32/libasan_preinit.o
- /usr/lib/gcc/x86_64-redhat-linux/8/crtbegin.o
- /usr/lib/gcc/x86_64-redhat-linux/8/crtbeginS.o
- /usr/lib/gcc/x86_64-redhat-linux/8/crtbeginT.o
- /usr/lib/gcc/x86_64-redhat-linux/8/crtend.o
- /usr/lib/gcc/x86_64-redhat-linux/8/crtendS.o
- /usr/lib/gcc/x86_64-redhat-linux/8/crtfastmath.o
- /usr/lib/gcc/x86_64-redhat-linux/8/crtoffloadbegin.o
- /usr/lib/gcc/x86_64-redhat-linux/8/crtoffloadend.o
- /usr/lib/gcc/x86_64-redhat-linux/8/crtoffloadtable.o
- /usr/lib/gcc/x86_64-redhat-linux/8/crtprec32.o
- /usr/lib/gcc/x86_64-redhat-linux/8/crtprec64.o
- /usr/lib/gcc/x86_64-redhat-linux/8/crtprec80.o
- /usr/lib/gcc/x86_64-redhat-linux/8/libasan_preinit.o
- /usr/lib/gcc/x86_64-redhat-linux/8/liblsan_preinit.o
- /usr/lib/gcc/x86_64-redhat-linux/8/libtsan_preinit.o

gconv files are data tables used for character conversion. These do not contain executable code.

- /usr/lib64/gconv/*

This is part of the bootloader and has special needs during boot.

- /usr/lib/grub/i386-pc/kernel.exec

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.

- /usr/libexec/os-prober/newns

Firmware bundled with the OS is loaded onto various hardware devices and does not execute on the stack, so stack smashing protections are not needed.

- /usr/lib/firmware/*

7 Acronyms

Table 17 – Acronyms

Acronym	Definition
AES	Advanced Encryption Standard
ASLR	Address Space Layout Randomization
CBC	Cipher Block Chaining
CC	Common Criteria
CMC	Certificate Management over CMS
CMS	Cryptographic Message Syntax
CRL	Certificate Revocation List
DNS	Domain Name System
DRBG	Deterministic Random Bit Generator
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
EKU	Extended Key Usage
EST	Enrollment over Secure Transport
FIPS	Federal Information Processing Standards
GCM	Galois Counter Mode
GPOS	General Purpose Operating System
HMAC	Hash-based Message Authentication Code
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IP	Internet Protocol
NIAP	Nation Information Assurance Partnership
NIST	National Institute of Standards and Technology
NTP	Network Time Protocol
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

Acronym	Definition
ST	Security Target
SWID	Software Identification
TOE	Target of Evaluation
TLS	Transport Layer Security
TSS	TOE Summary Specification