SUSE Linux Enterprise Micro 5.3
Security Target

Version:	**1.1**
Status:	**Released**
Last Update:	**2025-01-07**
Classification:	**Public**

Trademarks

SUSE and the SUSE logo are trademarks or registered trademarks of SUSE Linux Products GmbH in Germany, other countries, or both.

Linux is a registered trademark of Linus Torvalds.

UNIX is a registered trademark of The Open Group in the United States and other countries.

IBM, IBM logo, bladecenter, eServer, iSeries, OS/400, POWER9, POWER10, System x, System z, S390, xSeries, zSeries, zArchitecture, and z/VM are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

Intel, Xeon, and Pentium are trademarks of Intel Corporation in the United States, other countries, or both.

Other company, product, and service names may be trademarks or service marks of others.

Legal Notice

This document is provided AS IS with no express or implied warranties. Use the information in this document at your own risk.

This document may be reproduced and distributed only in its original entirety without revision.

Classification Note

Confidential Information (atsec confidential)

This classification level is for highly sensitive information. Access to "atsec confidential" information is limited to employees with a need to know. Information classified "atsec confidential" may be given to external persons with a need to know if they have signed an appropriate Non-Disclosure Agreement (NDA). Access to "atsec confidential" information stored on central IT systems must be controlled. Anyone obtaining information classified "atsec confidential" must apply the protection mechanisms necessary to ensure that the information cannot be obtained by anyone who does not have explicit authorization.

Information with this classification shall be clearly marked "atsec confidential" when it is in human-readable form. Electronic or other media storing information with this classification in unencrypted form shall be protected from any unauthorized access and shall be subject to specific handling procedures when they are reused or destroyed.

Copies of information classified "atsec confidential" shall only be made when necessary and must be strictly controlled.

Revision History

Revision	Date	Author(s)	Changes to Previous Revision
1.0	2024-12-13	SUSE supported by atsec consultants	Release
1.1	2025-01-07	SUSE supported by atsec consultants	Clarification on AppArmor

Table of Contents

List of Tables

1 Introduction

1.1 Security Target Identification

Title:	SUSE Linux Enterprise Micro 5.3 Security Target
Version:	1.1
Status:	Released
Date:	2025-01-07
Sponsor:	SUSE LLC
Developer:	SUSE LLC
Certification Body:	BSI
Certification ID:	BSI-DSZ-CC-1214
Keywords:	SLE Micro, operating system

1.2 TOE Identification

The TOE is SUSE Linux Enterprise Micro Version 5.3.

1.3 TOE Type

The TOE type is general purpose operating system.

1.4 TOE Overview

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.

The TOE is the SUSE Linux Enterprise Micro 5.3 general purpose operating system (GPOS). The TOE is a highly-configurable Linux-based operating system which has been developed to provide a good level of security as required in commercial environments. Details about the supported security functions are outlined in section 1.5.

The tested version of the TOE is:

- SLE Micro 5.3

1.5 TOE Description

This section provides a general description of the TOE, including physical boundaries, security functions, and relevant TOE documentation and references.

1.5.1 Physical Boundary

The Target of Evaluation is based on the following system software:

- SUSE Linux Enterprise Micro version 5.3

The TOE and its documentation are supplied a raw disk image distributed via the SUSE Portal. The TOE includes a package holding the additional user and administrator documentation.

In addition to the installation media, the following documentation is provided:

- Evaluated Configuration Guide published by SUSE, Version 4.0
- Manual pages for all applications, configuration files and system calls

Processor	microArch
Intel x86_64	Cascade Lake
AMD x86_64	AMD EPYC 3rd Generation
ARM 64 Bit	ARMv8.2-A

Table 1: Hardware platforms

The TOE also includes of the TOE documentation providing information for installing, configuring, and maintaining the evaluated configuration.

- SUSE Linux Enterprise Micro 5.3 Evaluated Configuration Guide, Version 4.0

1.5.2 TOE Security Functionality

The TOE provides the security functions conforms to the requirements defined in section 2.

1.5.2.1 Security audit

The TOE generates audit events for all start-up and shut-down functions, and all auditable events as specified by the requirements defined in section 2. Audit events are generated for the following audit functions:

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

Each audit record contains the date and time of the event, type of event, subject identity (if applicable), and outcome (success or failure) of the event.

1.5.2.2 Cryptographic support

The TOE includes the OpenSSL version 1.1.1 cryptographic libraries for performing userspace cryptographic operations. In addition, the Linux kernel crypto API performs the cryptographic operations performed by the kernel. In addition, the TOE uses software noise sources for entropy generation. The TOE implements TLSv1.2 for secure communications with remote servers.

The TOE implements SSHv2 for allowing secure remote administration.

1.5.2.3 User data protection

The TOE implements access controls which can be configured to prevent unprivileged users from accessing files and directories owned by other users. The configuration of the access control mechanism is left to the owner of the file system object.

1.5.2.4 Identification and authentication

All users including administrators must be authenticated to the TOE prior to carrying out any actions, including management operations. The TOE supports password-based authentication, authentication based on SSH-keys as well as X.509 certificate-based authentication. The TOE will lock out user accounts after a defined number of unsuccessful authentication attempts to that user account has been met.

1.5.2.5 Security Management

The TOE can perform management functions. The administrator has full access to carry-out all management functions offered by the TOE. The user is allowed a limited set of administrative operations for his own user account.

1.5.2.6 Protection of the TSF

The TOE implements the following protection of TSF data functions.

- Access controls
- Address space layout randomization (ASLR) with 11 bits (stack) and 28 bits (text segment start address) of entropy
- Stack buffer overflow protection
- Verification of integrity of the boot-chain
- Trusted software updates using digital signatures

1.5.2.7 TOE Access

The TOE displays an advisory warning message regarding unauthorized use of the OS prior to establishment of a user session.

1.5.2.8 Trusted Path/Channels

The TOE supports TLS v1.2 for trusted channel communications. The TOE uses TLS to securely communicate with the SUSE Customer Center. Applications may invoke the TOE-provided TLS to securely communicate with remote servers.

The TOE offers an SSH server which uses the SSHv2 protocol allowing remote administration.

1.5.3 TOE Operational Environment

The following environmental components interoperate with the TOE in the evaluated configuration.

Component	Description
Hardware platform	See Table 1
SUSE Customer Center	Server that allows the TOE to download updates

Table 2: TOE operational environment

1.5.4 Product Functionality Excluded from the Scope of the Evaluation

Additional mechanisms and functions that would interfere with the operation of the security functions are disallowed in the evaluated configuration and the Evaluation Configuration Guide provides instructions to the administrator on how to disable them. Note: TOE mechanism which provide additional restrictions to the above claimed security functions are allowed in the evaluated configuration. For example, the eCryptFS cryptographic file system provided with the TOE and permitted in the evaluated configuration even though they have not been subject to this evaluation. The eCryptFS provides further restrictions on, for example, the security function of discretionary access control mechanism for file system objects and therefore cannot breach the security functionality as the discretionary access control rules of the "lower" file system are still enforced. The following table enumerates mechanisms that are provided with the TOE but which are excluded from the evaluation:

Functions	Exclusion discussion
eCryptFS	eCryptFS is not allowed to be used in the evaluated configuration. The encryption capability provided with this file system is therefore unavailable to any user.
Ext4 file-based encryption	Ext4 file-based encryption is not allowed to be used in the evaluated configuration. The encryption capability provided with this file system is therefore unavailable to any user.
SMACK	The mandatory access control functionality offered by the SMACK LSM is not assessed by the evaluation and disabled in the evaluated configuration.
AppArmor	The mandatory access control functionality offered by the AppArmor LSM is not assessed by the evaluation and disabled in the evaluated configuration.
SSL / TLS tunnels	The TOE provides the stunnel application which can be used to establish SSL and TLS tunnels with remote peers. This application however was excluded from evaluation assessment.

Table 3: Non-evaluated functionalities

Note: Packages and mechanisms not covered with security claims and subsequent assessments during the evaluation or disabling the respective functionality in the evaluated configuration result from resource constraints during the evaluation but does not imply that the respective package or functionality is implemented insecurely.

2 CC Conformance Claim

This Security Target is CC Part 2 extended and CC Part 3 extended.

This Security Target claims conformance to the following Protection Profiles and PP packages:

- [OSPP]: Protection Profile for General Purpose Operating Systems. Version 4.2.1 as of 2019-04-22; exact conformance.
- [SSH]: Functional Package for Secure Shell (SSH). Version 1.0 as of 2021-05-13; exact conformance.

Common Criteria [CC] version 3.1 revision 5 is the basis for this conformance claim.

2.1 Protection Profile Tailoring and Additions

2.1.1 Protection Profile for General Purpose Operating Systems ([OSPP])

Table 4 contains the NIAP Technical Decisions (TDs) for this protection profile at the time of the evaluation and a statement of applicability to the evaluation.

NIAP TD	TD description	Applicable?
TD0715	Updates to FIA_X509_EXT.1 for Exception Processing and Test Conditions	Yes
TD0680	OS 4.2.1 Conformance Claims section updated to allow for MOD_WLAN_CLI_v1.0	Yes
TD0649	Conformance claims for OS PP v4.2.1	Yes
TD0630	FCS_COP.1 requirements for Secure Shell	Yes
TD0600	Conformance claim sections updated to allow for MOD_VPNC_V2.3	Yes
TD0578	SHA-1 is no longer mandatory	Yes
TD0501	Cryptographic selections and updates for OS PP	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 selection	Yes

Table 4: NIAP TDs for OSPP

2.1.2 Functional Package for Secure Shell (SSH) ([SSH])

Table 5 contains the NIAP Technical Decisions (TDs) for this PP-Module at the time of the evaluation and a statement of applicability to the evaluation.

NIAP TD	TD description	Applicable?
TD0777	TD0777: Clarification to Selections for Auditable Events for FCS_SSH_EXT.1	Yes
TD0732	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

Table 5: NIAP TDs for SSH

3 Security Problem Definition

3.1 Threat Environment

3.1.1 Threats countered by the TOE

T.NETWORK_ATTACK

PP Origin: *OSPP*

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

PP Origin: *OSPP*

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

PP Origin: *OSPP*

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

PP Origin: *OSPP*

An attacker may attempt to access data on the OS while having a limited amount of time with the physical device.

3.2 Assumptions

3.2.1 Intended usage of the TOE

A.PLATFORM

PP Origin: *OSPP*

The OS relies upon a trustworthy computing platform for its execution. This underlying platform is out of scope of this PP.

A.PROPER_USER

PP Origin: *OSPP*

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

PP Origin: *OSPP*

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

4 Security Objectives

4.1 Objectives for the TOE

O.ACCOUNTABILITY

PP Origin: *OSPP*

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

PP Origin: *OSPP*

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

PP Origin: *OSPP*

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

PP Origin: *OSPP*

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

PP Origin: *OSPP*

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 Objectives for the Operational Environment

OE.PLATFORM

PP Origin: *OSPP*

The OS relies on being installed on trusted hardware.

OE.PROPER_USER

PP Origin: *OSPP*

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

PP Origin: *OSPP*

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

4.3 Security Objectives Rationale

4.3.1 Coverage

The following table provides a mapping of TOE objectives to threats and policies, showing that each objective counters or enforces at least one threat or policy, respectively.

Objective	Threats / OSPs
O.ACCOUNTABILITY	T.NETWORK_ATTACK T.LOCAL_ATTACK
O.INTEGRITY	T.NETWORK_ATTACK T.LOCAL_ATTACK
O.MANAGEMENT	T.NETWORK_ATTACK T.NETWORK_EAVESDROP
O.PROTECTED_STORAGE	T.LIMITED_PHYSICAL_ACCESS
O.PROTECTED_COMMS	T.NETWORK_ATTACK T.NETWORK_EAVESDROP

Table 6: Mapping of security objectives to threats and policies

The following table provides a mapping of the objectives for the Operational Environment to assumptions, threats and policies, showing that each objective holds, counters or enforces at least one assumption, threat or policy, respectively.

Objective	Assumptions / Threats / OSPs
OE.PLATFORM	A.PLATFORM
OE.PROPER_USER	A.PROPER_USER
OE.PROPER_ADMIN	A.PROPER_ADMIN

Table 7: Mapping of security objectives for the Operational Environment to assumptions, threats and policies

4.3.2 Sufficiency

The following rationale provides justification that the security objectives are suitable to counter each individual threat and that each security objective tracing back to a threat, when achieved, actually contributes to the removal, diminishing or mitigation of that threat.

Threat	Rationale for security objectives
T.NETWORK_ATTACK	The threat T.NETWORK_ATTACK is countered by O.PROTECTED_COMMS as this provides for integrity of transmitted data.

Threat	Rationale for security objectives
	The threat T.NETWORK_ATTACK is countered by O.INTEGRITY as this provides for integrity of software that is installed onto the system from the network. The threat T.NETWORK_ATTACK is countered by O.MANAGEMENT as this provides for the ability to configure the OS to defend against network attack. The threat T.NETWORK_ATTACK is countered by O.ACCOUNTABILITY as this provides a mechanism for the OS to report behavior that may indicate a network attack has occurred.
T.NETWORK_EAVESDROP	The threat T.NETWORK_EAVESDROP is countered by O.PROTECTED_COMMS as this provides for confidentiality of transmitted data. The threat T.NETWORK_EAVESDROP is countered by O.MANAGEMENT as this provides for the ability to configure the OS to protect the confidentiality of its transmitted data.
T.LOCAL_ATTACK	The objective O.INTEGRITY protects against the use of mechanisms that weaken the TOE with regard to attack by other software on the platform. The objective O.ACCOUNTABILITY protects against local attacks by providing a mechanism to report behavior that may indicate a local attack is occurring or has occurred.
T.LIMITED_PHYSICAL_ACCESS	The objective O.PROTECTED_STORAGE protects against unauthorized attempts to access physical storage used by the TOE.

Table 8: Sufficiency of objectives countering threats

The following rationale provides justification that the security objectives for the environment are suitable to cover each individual assumption, that each security objective for the environment that traces back to an assumption about the environment of use of the TOE, when achieved, actually contributes to the environment achieving consistency with the assumption, and that if all security objectives for the environment that trace back to an assumption are achieved, the intended usage is supported.

Assumption	Rationale for security objectives
A.PLATFORM	The operational environment objective OE.PLATFORM is realized through A.PLATFORM.
A.PROPER_USER	The operational environment objective OE.PROPER_USER is realized through A.PROPER_USER.
A.PROPER_ADMIN	The operational environment objective OE.PROPER_ADMIN is realized through A.PROPER_ADMIN.

Table 9: Sufficiency of objectives holding assumptions

5 Extended Components Definition

The extended components definitions are defined in the documents specified in Section 2 "CC Conformance Claim".

6 Security Requirements

6.1 TOE Security Functional Requirements

The table below summarizes the SFRs for the TOE and the operations performed on the components according to CC part 1. Operations in the SFRs use the following convention:

- Iterations (Iter.) are identified by appending a suffix to the original SFR.
- Refinements (Ref.) added to the text are shown in *italic text*, deletions are shown as ~~strikethrough text~~.
- Assignments (Ass.) are shown in **bold text**.
- Selections (Sel.) are shown in **bold text**.

Security functional class	Security functional requirement	Base security functional component	Source	Operations			
				Iter.	Ref.	Ass.	Sel.
FAU - Security audit	FAU_GEN.1 Audit Data Generation (Refined)		OSPP	No	No	Yes	Yes
FCS - Cryptographic support	FCS_CKM.1 Cryptographic Key Generation (Refined)		OSPP	No	Yes	No	Yes
	FCS_CKM.2 Cryptographic Key Establishment (Refined)		OSPP	No	No	No	Yes
	FCS_CKM_EXT.4 Cryptographic Key Destruction		OSPP	No	No	No	Yes
	FCS_COP.1(1) Cryptographic Operation - Encryption/Decryption (Refined)	FCS_COP.1	OSPP	Yes	No	No	Yes
	FCS_COP.1(2) Cryptographic Operation - Hashing (Refined)	FCS_COP.1	OSPP	Yes	No	No	Yes
	FCS_COP.1(3) Cryptographic Operation - Signing (Refined)	FCS_COP.1	OSPP	Yes	Yes	No	Yes
	FCS_COP.1(4) Cryptographic Operation - Keyed-Hash Message Authentication (Refined)	FCS_COP.1	OSPP	Yes	No	Yes	Yes
	FCS_RBG_EXT.1 Random Bit Generation		OSPP	No	No	No	Yes
	FCS_STO_EXT.1 Storage of Sensitive Data		OSPP	No	No	No	No
	FCS_TLSC_EXT.1 TLS Client Protocol		OSPP	No	No	No	Yes
	FCS_TLSC_EXT.2 TLS Client Protocol		OSPP	No	No	No	Yes
	FCS_TLSC_EXT.4 TLS Client Protocol		OSPP	No	No	No	No
	FCS_SSH_EXT.1 SSH Protocol		SSH	No	No	Yes	Yes

Security functional class	Security functional requirement	Base security functional component	Source	Operations			
				Iter.	Ref.	Ass.	Sel.
	FCS_SSHC_EXT.1 SSH Protocol - Client		SSH	No	No	No	Yes
	FCS_SSHS_EXT.1 SSH Protocol - Server		SSH	No	No	No	Yes
FDP - User data protection	FDP_ACF_EXT.1 Access Controls for Protecting User Data		OSPP	No	No	No	No
FIA - Identification and authentication	FIA_AFL.1 Authentication failure handling (Refined)		OSPP	No	No	Yes	Yes
	FIA_UAU.5 Multiple Authentication Mechanisms (Refined)		OSPP	No	No	Yes	Yes
	FIA_X509_EXT.1 X.509 Certificate Validation		OSPP	No	No	Yes	Yes
	FIA_X509_EXT.2 X.509 Certificate Authentication		OSPP	No	No	No	Yes
FMT - Security management	FMT_MOF_EXT.1 Management of security functions behavior		OSPP	No	No	No	No
	FMT_SMF_EXT.1 Specification of Management Functions		OSPP	No	No	Yes	Yes
FPT - Protection of the TSF	FPT_ACF_EXT.1 Access controls		OSPP	No	No	Yes	No
	FPT_ASLR_EXT.1 Address Space Layout Randomization		OSPP	No	No	Yes	Yes
	FPT_SBOP_EXT.1 Stack Buffer Overflow Protection		OSPP	No	No	No	Yes
	FPT_TST_EXT.1 Boot Integrity		OSPP	Yes	No	No	Yes
	FPT_TUD_EXT.1 Trusted Update		OSPP	No	No	No	Yes
	FPT_TUD_EXT.2 Trusted Update for Application Software		OSPP	No	No	No	No
FTA - TOE access	FTA_TAB.1 Default TOE access banners		OSPP	No	No	No	No
FTP - Trusted path/channels	FTP_ITC_EXT.1 Trusted channel communication		OSPP	No	No	Yes	Yes
	FTP_TRP.1 Trusted Path		OSPP	No	No	No	Yes

Table 10: SFRs for the TOE

6.1.1 Security audit (FAU)

6.1.1.1 FAU_GEN.1 Audit Data Generation (Refined)

PP Origin: *OSPP*

PP Origin: *SSH*

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);
- **Administrator or root-level access events (Success/Failure)**
- **FCS_SSH_EXT.1: None**

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, **none**.

TSS Link: *TSS for FAU_GEN.1*

6.1.2 Cryptographic support (FCS)

6.1.2.1 FCS_CKM.1 Cryptographic Key Generation (Refined)

PP Origin: *OSPP*

Applied TDs: *TD0501*

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~~ *3072-bit* or greater that meet the following: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.3**
- **ECC schemes using "NIST curves" P-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"**

TSS Link: *TSS for FCS_CKM.1*

6.1.2.2 FCS_CKM.2 Cryptographic Key Establishment (Refined)

PP Origin: *OSPP*

Applied TDs: *TD0501*

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

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

TSS Link: *TSS for FCS_CKM.2*

6.1.2.3 FCS_CKM_EXT.4 Cryptographic Key Destruction

PP Origin: *OSPP*

Applied TDs: *TD0365*

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**
  - **destruction of all key encrypting keys (KEKs) protecting the target key according to FCS_CKM_EXT.4.1, where none of the KEKs protecting the target key are derived**
  - **the invocation of an interface provided by the underlying platform that**
    - **instructs the underlying platform to destroy the abstraction that represents the key**

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

TSS Link: *TSS for FCS_CKM_EXT.4*

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

PP Origin: *OSPP*

Applied TDs: *TD0630*

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)**
- **AES-XTS (as defined in NIST SP 800-38E)**

and

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

and cryptographic key sizes

- **128-bit**
- **256-bit.**

TSS Link: *TSS for FCS_COP.1(1)*

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

PP Origin: *OSPP*

Applied TDs: *TD0578*

FCS_COP.1.1(2) The OS shall perform cryptographic hashing services in accordance with a specified cryptographic algorithm

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

and message digest sizes

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

that meet the following: FIPS Pub 180-4.

TSS Link: *TSS for FCS_COP.1(2)*

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

PP Origin: *OSPP*

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~~ *3072-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**

TSS Link: *TSS for FCS_COP.1(3)*

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

PP Origin: *OSPP*

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.

TSS Link: *TSS for FCS_COP.1(4)*

6.1.2.8 FCS_RBG_EXT.1 Random Bit Generation

PP Origin: *OSPP*

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

TSS Link: *TSS for FCS_RBG_EXT.1*

Application Note CTR DRBG:

For the German Schema, the SFR is "translated" into an AIS 20/31 compliant SFR following the FCS_RNG.1 definition. Therefore, this application note states the SFR as part of this application note:

FCS_RNG.1.1: The TSF shall provide a deterministic random number generator conforming to SP800-90A CTR_DRBG with AES-256 core using a derivation function without prediction resistance that implements:

- *a) DRG2.1: If initialized with a random seed using high-resolution time stamps of block device access events, human interface device events and interrupt events as seed source, the internal state of the RNG shall have a minentropy of 256 bits.*
- *b) DRG2.2: The DRNG provides forward secrecy.*
- *c) DRG2.3: The DRNG provides backward secrecy.*

The TSF shall provide random numbers that meet:

- *a) DRG.2.4: The RNG is initialized with a random seed of 384 bits, is reseeded after at most 2**48 generate requests with 256 bits, and has the output property such that 2**19 strings of bit length 128 are mutually different with probability of greater than 1-2**-10.*
- *b) DRG.2.5: Statistical test suites cannot practically distinguish the random numbers from output sequences of an ideal RNG. The random numbers must pass test procedure A.*

6.1.2.9 FCS_STO_EXT.1 Storage of Sensitive Data

PP Origin: *OSPP*

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.

TSS Link: *TSS for FCS_STO_EXT.1*

6.1.2.10 FCS_TLSC_EXT.1 TLS Client Protocol

PP Origin: *OSPP*

Applied TDs: *TD0441*

FCS_TLSC_EXT.1.1 The product 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**

.

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

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

- **with no exceptions.**

TSS Link: *TSS for FCS_TLSC_EXT.1*

6.1.2.11 FCS_TLSC_EXT.2 TLS Client Protocol

PP Origin: *OSPP*

FCS_TLSC_EXT.2.1 The product shall present the Supported Groups Extension in the Client Hello with the supported groups

- **secp256r1**
- **secp384r1**
- **secp521r1**

TSS Link: *TSS for FCS_TLSC_EXT.2*

6.1.2.12 FCS_TLSC_EXT.4 TLS Client Protocol

PP Origin: *OSPP*

FCS_TLSC_EXT.4.1 The product shall support mutual authentication using X.509v3 certificates.

TSS Link: *TSS for FCS_TLSC_EXT.4*

6.1.2.13 FCS_SSH_EXT.1 SSH Protocol

PP Origin: *SSH*

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, **4256, 5647, 5656, 6668, 8268, 8332, 8709** and no other standard.

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

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

and no other methods.

FCS_SSH_EXT.1.3 The TSF shall ensure that, as described in RFC 4253, packets greater than **2^18 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-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-group16-sha512 (RFC 8268)**
- **diffie-hellman-group18-sha512 (RFC 8268)**
- **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 4253 (Section 7.2)**
- **RFC 5656 (Section 4)**

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

FCS_SSH_EXT.1.8 The TSF shall ensure that

- **a rekey of the session keys**

occurs when any of the following thresholds are met:

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

Application Note: *If the TOE-attempted rekey is not accepted or completed by the remote peer, the connection is terminated.*

TSS Link: *TSS for FCS_SSH_EXT.1*

6.1.2.14 FCS_SSHC_EXT.1 SSH Protocol - Client

PP Origin: *SSH*

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

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

as described in RFC 4251 section 4.1.

TSS Link: *TSS for FCS_SSH_EXT.1*

6.1.2.15 FCS_SSHS_EXT.1 SSH Protocol - Server

PP Origin: *SSH*

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

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

.

TSS Link: *TSS for FCS_SSH_EXT.1*

6.1.3 User data protection (FDP)

6.1.3.1 FDP_ACF_EXT.1 Access Controls for Protecting User Data

PP Origin: *OSPP*

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.

TSS Link: *TSS for FDP_ACF_EXT.1*

6.1.4 Identification and authentication (FIA)

6.1.4.1 FIA_AFL.1 Authentication failure handling (Refined)

PP Origin: *OSPP*

FIA_AFL.1.1 The OS shall detect when **an administrator configurable positive integer within 1 and 2^32 - 1** 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**.

TSS Link: *TSS for FIA_AFL.1*

6.1.4.2 FIA_UAU.5 Multiple Authentication Mechanisms (Refined)

PP Origin: *OSPP*

Applied TDs: *TD0649*

FIA_UAU.5.1 The OS shall provide the following authentication mechanisms

- **authentication based on username and password**
- **for use in SSH only, SSH public key-based authentication as specified by the Functional Package for Secure Shell**

to support user authentication.

FIA_UAU.5.2 The OS shall authenticate any user's claimed identity according to the

- **Authentication based on username and password: is performed for SSH password-based as well as console login requests with credentials stored by the OS.**
- **Authentication based on SSH keys: is performed SSH key-based login requests with SSH keys stored by the OS.**

TSS Link: *TSS for FIA_UAU.5*

6.1.4.3 FIA_X509_EXT.1 X.509 Certificate Validation

PP Origin: *OSPP*

Applied TDs: *TD0715*

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

- RFC 5280 certificate validation and certificate path validation
- The certificate path must terminate with a trusted CA certificate
- The OS shall validate a certificate path by ensuring the presence of the basicConstraints extension, that the CA flag is set to TRUE for all CA certificates, and that any path constraints are met.
- The TSF shall validate that any CA certificate includes "Certificate Signing" as a purpose the key usage field
- The OS shall validate the revocation status of the certificate using **CRL as specified in RFC 8603, an OCSP TLS Status Request Extension (OCSP stapling) as specified in RFC 6066** with **the exception of the secure boot process where no viable network link is yet present**
- 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.

TSS Link: *TSS for FIA_X509_EXT.1*

6.1.4.4 FIA_X509_EXT.2 X.509 Certificate Authentication

PP Origin: *OSPP*

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.

TSS Link: *TSS for FIA_X509_EXT.2*

6.1.5 Security management (FMT)

6.1.5.1 FMT_MOF_EXT.1 Management of security functions behavior

PP Origin: *OSPP*

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.

TSS Link: *TSS for FMT_MOF_EXT.1*

6.1.5.2 FMT_SMF_EXT.1 Specification of Management Functions

PP Origin: *OSPP*

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

#	Management Function	Administrator	User
1	Enable/disable **screen lock, session timeout**	M	X
2	Configure **screen lock, session** inactivity timeout	M	X
3	Import keys/secrets into the secure key storage	X	X
4	Configure local audit storage capacity	X	-
5	Configure minimum password length	X	-
6	Configure minimum number of special characters in password	X	-
7	Configure minimum number of numeric characters in password	X	-
8	Configure minimum number of uppercase characters in password	X	-
9	Configure minimum number of lowercase characters in password	X	-
10	Configure lockout policy for unsuccessful authentication attempts through **limiting number of attempts during a time period**	X	-
11	Configure host-based firewall	X	-
12	Configure name/address of directory server with which to bind	-	-
13	Configure name/address of remote management server from which to receive management settings	-	-
14	Configure name/address of audit/logging server to which to send audit/logging records	X	-
15	Configure audit rules	X	-
16	Configure name/address of network time server	X	-
17	Enable/disable automatic software update	X	-
18	Configure Wi-Fi interface	X	-
19	Enable/disable Bluetooth interface	-	-
20	Enable/disable **no other external interfaces**	-	-

#	Management Function	Administrator	User
21	**No other management functions to be provided by the TSF**	-	-

Table 11: Management functions (OSPP)

Application Note: *M—Mandatory support by the specified role. X—Supported by the specified role. Grey/Hyphen—Not supported by the specified role.*

TSS Link: *TSS for FMT_SMF_EXT.1*

6.1.6 Protection of the TSF (FPT)

6.1.6.1 FPT_ACF_EXT.1 Access controls

PP Origin: *OSPP*

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

a) Kernel and its drivers/modules
b) Security audit logs
c) Shared libraries
d) System executables
e) System configuration files
f) **TSF-data including permission bits, ACLs**
g) **Third-party applications**

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

a) Security audit logs
b) System-wide credential repositories
c) **no other objects**

TSS Link: *TSS for FPT_ACF_EXT.1*

6.1.6.2 FPT_ASLR_EXT.1 Address Space Layout Randomization

PP Origin: *OSPP*

FPT_ASLR_EXT.1.1 The OS shall always randomize process address space memory locations with **11 (stack memory), 28 (start address of text segments)** bits of entropy except for **no exceptions**.

TSS Link: *TSS for FPT_ASLR_EXT.1*

6.1.6.3 FPT_SBOP_EXT.1 Stack Buffer Overflow Protection

PP Origin: *OSPP*

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

TSS Link: *TSS for FPT_SBOP_EXT.1*

6.1.6.4 FPT_TST_EXT.1 Boot Integrity

PP Origin: *OSPP*

Applied TDs: *TD0493*

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 an X509 certificate with hardware-based protection**

TSS Link: *TSS for FPT_TST_EXT.1*

6.1.6.5 FPT_TUD_EXT.1 Trusted Update

PP Origin: *OSPP*

Applied TDs: *TD0386*

Applied TDs: *TD0463*

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).

TSS Link: *TSS for FPT_TUD_EXT.1*

6.1.6.6 FPT_TUD_EXT.2 Trusted Update for Application Software

PP Origin: *OSPP*

Applied TDs: *TD0463*

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.

TSS Link: *TSS for FPT_TUD_EXT.2*

6.1.7 TOE access (FTA)

6.1.7.1 FTA_TAB.1 Default TOE access banners

PP Origin: *OSPP*

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

TSS Link: *TSS for FTA_TAB.1*

6.1.8 Trusted path/channels (FTP)

6.1.8.1 FTP_ITC_EXT.1 Trusted channel communication

PP Origin: *OSPP*

Applied TDs: *TD0649*

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: **management server, update server, application initiated TLS, SSH peer** 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.

TSS Link: *TSS for FTP_ITC_EXT.1*

6.1.8.2 FTP_TRP.1 Trusted Path

PP Origin: *OSPP*

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** 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.

TSS Link: *TSS for FTP_TRP.1*

6.2 Security Functional Requirements Rationale

6.2.1 Coverage

The following table provides a mapping of SFR to the security objectives, showing that each security functional requirement addresses at least one security objective.

Security functional requirements	Objectives
FAU_GEN.1	O.ACCOUNTABILITY
FCS_CKM.1	O.PROTECTED_COMMS
FCS_CKM.2	O.PROTECTED_COMMS
FCS_CKM_EXT.4	O.PROTECTED_COMMS
FCS_COP.1(1)	O.PROTECTED_COMMS, O.PROTECTED_STORAGE
FCS_COP.1(2)	O.INTEGRITY, O.PROTECTED_COMMS
FCS_COP.1(3)	O.INTEGRITY, O.PROTECTED_COMMS
FCS_COP.1(4)	O.INTEGRITY, O.PROTECTED_COMMS
FCS_RBG_EXT.1	O.PROTECTED_COMMS, O.PROTECTED_STORAGE
FCS_STO_EXT.1	O.PROTECTED_STORAGE
FCS_TLSC_EXT.1	O.PROTECTED_COMMS

Security functional requirements	Objectives
FCS_TLSC_EXT.2	O.PROTECTED_COMMS
FCS_TLSC_EXT.4	O.PROTECTED_COMMS
FCS_SSH_EXT.1	O.PROTECTED_COMMS
FCS_SSHC_EXT.1	O.PROTECTED_COMMS
FCS_SSHS_EXT.1	O.PROTECTED_COMMS
FDP_ACF_EXT.1	O.PROTECTED_STORAGE
FIA_AFL.1	O.INTEGRITY
FIA_UAU.5	O.INTEGRITY
FIA_X509_EXT.1	O.INTEGRITY, O.PROTECTED_COMMS
FIA_X509_EXT.2	O.PROTECTED_COMMS
FMT_MOF_EXT.1	O.MANAGEMENT
FMT_SMF_EXT.1	O.MANAGEMENT
FPT_ACF_EXT.1	O.INTEGRITY
FPT_ASLR_EXT.1	O.INTEGRITY
FPT_SBOP_EXT.1	O.INTEGRITY
FPT_TST_EXT.1	O.INTEGRITY
FPT_TUD_EXT.1	O.INTEGRITY
FPT_TUD_EXT.2	O.INTEGRITY
FTA_TAB.1	O.MANAGEMENT
FTP_ITC_EXT.1	O.ACCOUNTABILITY, O.INTEGRITY, O.PROTECTED_COMMS
FTP_TRP.1	O.MANAGEMENT

Table 12: Mapping of security functional requirements to security objectives

6.2.2 Sufficiency

The following rationale provides justification for each security objective for the TOE, showing that the security functional requirements are suitable to meet and achieve the security objectives.

Security objectives	Rationale
O.ACCOUNTABILITY	[OSPP] FAU_GEN.1: Supports the objective by requiring that critical event information be gathered by the TOE. [OSPP] FTP_ITC_EXT.1: Supports the objective by ensuring that audit information can be securely transmitted to remote systems for analysis.

Security objectives	Rationale
O.INTEGRITY	[OSPP] FPT_SBOP_EXT.1, FPT_ASLR_EXT.1: Supports the objective by requiring that OS applications be hardened against buffer overflow attacks. [OSPP] FPT_TUD_EXT.1: Supports the objective by requiring that the OS be able to check for critical updates. [OSPP] FPT_TUD_EXT.2: Supports the objective by requiring that the OS verify updates before applying them. [OSPP] FCS_COP.1(2): Supports the objective by requiring the TSF to implement hash algorithms that are used in support of protected communications. [OSPP] FCS_COP.1(3): Supports the objective by requiring the TSF to implement digital signature algorithms that are used in support of protected communications. [OSPP] FCS_COP.1(4): Supports the objective by requiring the TSF to implement HMAC algorithms that are used in support of protected communications. [OSPP] FPT_ACF_EXT.1: Supports the objective by requiring the TSF restrict unprivileged users from changing critical components. [OSPP] FIA_X509_EXT.1: Supports the objective by requiring the TSF to validate certificates using industry standards. [OSPP] FPT_TST_EXT.1: Supports the objective by requiring the TSF to verify executable code critical to its operation. [OSPP] FTP_ITC_EXT.1: Supports the objective by requiring the OS to provide a trusted channel for critical communication. [OSPP] FIA_AFL.1: Supports the objective by requiring the TSF to respond accordingly when the number of failed authentication attempts reaches a specified threshold. [OSPP] FIA_UAU.5: Supports the objective by requiring the OS to provide standard authentication mechanisms.
O.MANAGEMENT	[OSPP] FMT_MOF_EXT.1: Supports this objective by requiring the TOE to restrict the ability to perform certain management functions to a privileged user. [OSPP] FMT_SMF_EXT.1: Supports this objective by requiring the TOE to implement specific management functions. [OSPP] FTA_TAB.1: Supports this objective by requiring the TOE to implement a trusted path between the itself and users. [OSPP] FTP_TRP.1: Supports this objective by requiring a trusted path between users and the OS.
O.PROTECTED_STORAGE	[OSPP] FCS_STO_EXT.1: Supports this objective by requiring the OS to provide encrypted storage. [OSPP] FCS_RBG_EXT.1: Supports this objective by requiring the OS to generate random bits according to industry standards. [OSPP] FCS_COP.1(1): Supports this objective requiring the OS to perform encryption according to industry stands. [OSPP] FDP_ACF_EXT.1: Supports this objective by requiring the OS to implement access controls.

Security objectives	Rationale
O.PROTECTED_COMMS	[OSPP] FCS_RBG_EXT.1: Supports this objective by requiring the OS to generate random bits according to industry standards. [OSPP] FCS_CKM.1: Supports this objective by requiring the TSF to generate asymmetric cryptographic keys to industry standards. [OSPP] FCS_CKM.2: Supports this objective by requiring the TSF to perform key establishment according to industry standards. [OSPP] FCS_CKM_EXT.4: Supports this objective by requiring the TSF to destroy key material according to industry standards. [OSPP] FCS_COP.1(1): Supports this objective by requiring the TSF to encrypt data according to industry standards. [OSPP] FCS_COP.1(2): Supports this objective by requiring the TSF to hash data according to industry standards. [OSPP] FCS_COP.1(3): Supports this objective by requiring the TSF to cryptographically sign data according to industry standards. [OSPP] FCS_COP.1(4): Supports this objective by requiring the TSF to perform keyed hashes according to industry standards. [OSPP] FIA_X509_EXT.1: Supports the objective by requiring the TSF to validate certificates using industry standards. [OSPP] FIA_X509_EXT.2: Supports this objective by requiring the TSF to validate TLS and related encrypted connections with x509 certificates. [OSPP] FTP_ITC_EXT.1: Supports the objective by requiring the OS to provide a trusted channel for critical communication. [ST] FCS_TLSC_EXT.1, FCS_TLSC_EXT.2, FCS_TLSC_EXT.4, FCS_SSH_EXT.1, FCS_SSHC_EXT.1, FCS_SSHS_EXT.1 define the ability of the TOE to act as a TLS client and SSH client/server as a method of enforcing protected communications.

Table 13: Security objectives for the TOE rationale

6.3 Security Assurance Requirements

The security assurance requirements (SARs) for the TOE are defined in the OSPP protection profile; and defined in CC assurance package.

6.3.1 ALC Life-cycle support

6.3.1.1 ALC_TSU_EXT.1 Timely Security Updates

Developer action elements:

ALC_TSU_EXT.1.1D The developer shall provide a description in the TSS of how timely security updates are made to the OS.

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

Content and presentation elements:

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

ALC_TSU_EXT.1.2C The description shall include the mechanisms publicly available for reporting security issues pertaining to the OS.

Evaluator action elements:

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

6.4 Security Assurance Requirements Rationale

SAR rationales are provided by the PPs to which this ST conforms. Section 2 contains the list of PPs.

7 TOE Summary Specification

7.1 TSS Security Assurance Evaluation Activity

7.1.1 Timely security updates (ALC_TSU_EXT.1)

The entire TOE (and in fact the entire SLES distribution) is subject to an extensive update process. The update process starts when SUSE is informed about defects. Depending on the severity (security incidents are considered to be severe), fixes are developed, tested and released with updated RPM packages.

Guaranteed response times depend on the selected service level agreement which is outlined in https://www.suse.com/support/handbook/.

SUSE generally does not disclose, discuss, or confirm security issues until a full investigation is complete and any necessary patches or releases are available. Further details about the security release process can be obtained at https://www.suse.com/support/security/ Once an issue has been confirmed and a patch has been made available, references containing technical details on the patches are made available and Common Vulnerabilities and Exposures (CVEs), etc. are released.

SUSE distributes information about security issues in its products through its "SUSE Update Advisory" page. (https://www.suse.com/support/update/)

The entire update process is handled by SUSE and covers all packages shipped as part of the SLES distribution from which the TOE is a subset.

Potential security vulnerabilities can be reported by following the procedures on the "SUSE Security Contacts" page (https://www.suse.com/support/security/contact/). This includes sending an email to "security@suse.com" or "security@suse.de" and includes the ability to encrypt information using the SUSE Security Team PGP key.

7.2 TOE Security Functionality

7.2.1 Audit

7.2.1.1 FAU_GEN.1 Audit Data Generation (Refined)

PP Origin: *OSPP*

PP Origin: *SSH*

SFR Link: *FAU_GEN.1*

Audit events are generated for the following audit functions.

- Start-up and shut-down of the audit functions
- 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)
- Administrator or root-level access events (Success/Failure)

Each audit record contains the following information.

- Date and time
- Type of event
- Subject identity (if applicable)

- Outcome (success or failure)

The Lightweight Audit Framework (LAF) is designed to be an audit system for Linux compliant with the requirements from Common Criteria. LAF is able to intercept all system calls as well as retrieving audit log entries from privileged user space applications. The subsystem allows configuring the events to be actually audited from the set of all events that are possible to be audited. Those events are configured in a specific configuration file and then the kernel is notified to build its own internal structure for the events to be audited.

Audit functionality

The Linux kernel implements the core of the LAF functionality. It gathers all audit events, analyzes these events based on the audit rules and forwards the audit events that are requested to be audited to the audit daemon executing in user space.

Audit events are generated in various places of the kernel. In addition, a user space application can create audit records which needs to be fed to the kernel for further processing.

The audit functionality of the Linux kernel is configured by user space applications which communicate with the kernel using a specific netlink communication channel. This netlink channel is also to be used by applications that want to send an audit event to the kernel.

The kernel netlink interface is usable only by applications possessing the following capabilities:

- CAP_AUDIT_CONTROL: Performing management operations like adding or deleting audit rules, setting or getting auditing parameters;
- CAP_AUDIT_WRITE: Submitting audit records to the kernel which in turn forwards the audit records to the audit daemon.

Based on the audit rules, the kernel decides whether an audit event is discarded or to be sent to the user space audit daemon for storing it in the audit trail. The kernel sends the message to the audit daemon again using the above mentioned netlink communication channel. The audit daemon writes the audit records to the audit trail. An internal queuing mechanism is used for this purpose. When the queue does not have sufficient space to hold an audit record the TOE switches into single user mode, is halted, all processes are stopped that generate audit records, or the audit daemon executes an administrator-specified notification action depending on the configuration of the audit daemon. This ensures that audit records do not get lost due to resource shortage and the administrator can backup and clear the audit trail to free disk space for new audit logs.

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.

The system administrator can define the events to be audited from the overall events that the Lightweight Audit Framework using simple filter expressions. This allows for a flexible definition of the events to be audited and the conditions under which events are audited. The system administrator is also able to define a set of user IDs for which auditing is active or alternatively a set of user IDs that are not audited.

The system administrator can select files to be audited by adding them to a watch list that is loaded into the kernel.

The audit trail is stored in files that are readable by the root user only.

Audit trail

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 the audit record was generated
- Audit ID: unique numerical event identifier
- 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 ID: the effective user 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)

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 audit ID to permit easy correlation.

The audit trail is stored in ASCII text. The TOE provides tools for managing ASCII files that can be used for post-processing of audit data. These tools include:

- less - reads the ASCII audit data
- ausearch - allows selective extraction of records from the audit trail using defined selection criteria
- sort - The audit records are listed in chronological order by default. The sort utility can be used together with ausearch to use a different sorting order.

The audit trail is stored in files which are accessible by root only.

7.2.2 Cryptography

The security features that use cryptography in this ST are the following.

- SSH
- Secure boot and integrity protection
- TLS client
- Trusted update
- Disk Encryption

The cryptographic modules used to implement the above security features are the following.

- OpenSSL (user space)
- Linux Kernel Crypto API (kernel space)
- libgcrypt (support for GPG)

Also the disk encryption is performed by the kernel using the kernel crypto API.

The integrity verification of updates is performed using GPG using libgcrypt.

All other mechanisms rely on OpenSSL for the cryptographic primitives.

Table 14 lists the algorithms discussed in the following subsections.

SFR	Algorithm	Capabilities	Usage
FCS_CKM.1	RSA KeyGen	3072, 4096	TLS client mutual authentication SSH mutual authentication
	ECDSA KeyGen	P-384, P-521, P-256	TLS client key establishment and mutual authentication SSH key establishment and mutual authentication

SFR	Algorithm	Capabilities	Usage
	FFC schema KeyGen	safe primes compliant to SP800-56A rev. 3	TLS client key establishment SSH key establishment
FCS_CKM.2	ECC Key Establishment (KAS-ECC)	P-384, P-521	TLS client key establishment SSH key establishment
	FFC Key Establishment (KAS-FFC)	safe primes compliant to SP800-56A rev. 3	TLS client key establishment SSH key establishment
FCS_COP.1(1)	AES-CBC, AES-CTR, AES-GCM, AES-XTS	128-bit, 256-bit (which implies 512 bits for AES-XTS)	TLS client SSH client / server Disk encryption
FCS_COP.1(2)	SHA-256, SHA-384, SHA-512		Secure boot TLS client Trusted update SSH client / server
FCS_COP.1(3)	RSA SigGen/SigVer	3072, 4096 with: SHA-256, SHA-384, SHA-512	Secure boot TLS client Trusted update SSH client / server
	ECDSA SigGen/SigVer	P-256, P-384, P-521 with: SHA-256, SHA-384, SHA-512	TLS client SSH client / server
FCS_COP.1(4)	HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512		TLS client SSH client / server
FCS_RBG_EXT.1	CTR_DRBG	AES	TLS client SSH client / server

Table 14: Cryptographic algorithm table

7.2.2.1 FCS_CKM.1 Cryptographic Key Generation

PP Origin: *OSPP*

SFR Link: *FCS_CKM.1*

The TOE supports generation of 3072-bit, and 4096-bit RSA keys conforming to FIPS PUB 186-4 Digital Signature Standard (DSS), Appendix B.3. The TOE supports the generation of RSA keys for use in mutual authentication of TLS, SSH sessions.

The TOE supports NIST curves P-256, P-384, and P-521 for key generation conforming to FIPS PUB 186-4 Digital Signature Standard (DSS)”, Appendix B.4. TLS, SSH sessions use these curves for ECDH key establishment and for ECDSA-based client authentication.

The TOE supports FFC-DH using Safe-Primes as defined by the NIST Special Publication 800-56A Revision 3, “Recommendation for Pair-Wise Key Establishment Schemes".

Please refer to Table 14 for details.

7.2.2.2 FCS_CKM.2 Cryptographic Key Establishment

PP Origin: *OSPP*

SFR Link: *FCS_CKM.2*

The TOE supports cryptographic key establishment using the following schemes.

- Elliptic curve-based key establishment with NIST curves P-256, P-384, and P-521 as specified in NIST SP 800-56A rev 3, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography"
- Finite field-based key establishment schemes that meets NIST Special Publication 800-56A Revision 3, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography" using safe primes as defined in SP800-56A rev 3.

Elliptic curve-based key establishment is used when ECDHE ciphersuites are negotiated for TLS sessions. FFC-based key establishment is used when DHE ciphersuites are negotiated for TLS sessions. For SSH support, ECDH-based key establishment is supported.

For SSH support, DH-based key establishment is supported.

Please refer to Table 14 for details.

7.2.2.3 FCS_CKM_EXT.4 Cryptographic Key Destruction

PP Origin: *OSPP*

SFR Link: *FCS_CKM_EXT.4*

Ephemeral cryptographic key material is held in RAM and is cleared oas soon as it is not required any more by overwriting the memory location with zeros.

The following keys are managed by OpenSSL on behalf of TLS and SSH:

- Ephemeral symmetric keys: The keys are derived during the TLS/SSH handshake from the Diffie-Hellman operation and stay in memory until rekey or destruction of the connection.
- Ephemeral MAC keys: The keys are derived during the TLS/SSH handshake from the Diffie-Hellman operation and stay in memory until rekey or destruction of the connection.
- Ephemeral DH/ECDH keys: The keys are generated by OpenSSL using its DRBG during the TLS/SSH handshake from the Diffie-Hellman operation and stay in memory until the handshake is completed.
- Ephemeral representation of asymmetric keys: The keys are read from files during startup of the application handling the TLS/SSH connection and stay in memory until the application terminates.

All ephemeral keys are held by OpenSSL in cipher handles which contains the memory buffer holding the keys. When the cipher handle is released, the key memory is zeroized. Also the removal of power to the RAM will zeroize the key material.

The following keys are managed by the Linux kernel crypto API on behalf the disk encryption mechanism of dm-crypt:

- Ephemeral symmetric key: The master volume key is stored wrapped in the LUKS header. That header is read by the tool cryptsetup that uses libgcrypt as its cryptographic library. The LUKS header is unwrapped using a passcode provided by the administrator. The unwrapped master volume key is injected into the kernel crypto API which wraps the key in a cipher handle that contains the memory buffer holding the key. When the

dm-crypt partition is unmounted, the Linux kernel crypto API cipher handle is deallocated which also zeroizes the memory buffer holding the master volume key. After cryptsetup injected the master volume key, the passcode, the key derived from it using PBKDF2 as well as its copy of the master volume key is overwritten with zeros. Also the removal of power to the RAM will zeroize the key material.

For non-volatile memory, the life of a TLS certificate / key and SSH key pairs is indefinite. A user or administrator can manually destroy them. To erase long-term key material held in files the TOE provides the tool fstrim. After a deletion of a file with sensitive data, this tool uses the SSD TRIM command to inform the SSD to discard unused blocks bypassing wear leveling. In addition, the tool shred is available that overwrites files multiple times with random data. This tool can be used for HDD to delete data.

As the LUKS header only holds the wrapped master volume key, it is not subject to clearing requirements. The KEK wrapping the master volume key as well as the user passphrase from which the KEK is derived are held in volatile store and thus are subject to the associated clearing mechanism. The unwrapped master volume key is also only held in volatile store subject to the same clearing mechanism. With the destruction of the master volume key that is used to protect the encrypted disk, all data on that disk are cryptographically erased.

7.2.2.4 FCS_COP.1(1) Cryptographic Operation - Encryption/Decryption

PP Origin: *OSPP*

SFR Link: *FCS_COP.1(1)*

The TOE supports AES encryption using 256-bit keys in the following modes.

- CBC as specified in NIST SP 800-38A (for TLS, SSH)
- CTR as specified in NIST SP 800-38A (SSH)
- GCM as specified in NIST SP 800-38D (for TLS, SSH)
- XTS as specified in NIST SP 800-38E (for disk encryption)

Please refer to Table 14 for details.

7.2.2.5 FCS_COP.1(2) Cryptographic Operation - Hashing

PP Origin: *OSPP*

SFR Link: *FCS_COP.1(2)*

The TOE supports cryptographic hashing services conforming to FIPS Pub 180-4. The hashing algorithms are used for signature services and HMAC services.

The following hashing algorithms supported: SHA-256, SHA-384, and SHA-512. The message digest sizes supported are: 256 bits, 384 bits, and 512 bits.

Please refer to Table 14 for details.

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

PP Origin: *OSPP*

SFR Link: *FCS_COP.1(3)*

The TOE provides cryptographic signature generation and verification in accordance with the following cryptographic algorithms.

- RSA digital signature algorithm conforming to FIPS Pub 186-4, "Digital Signature Standard (DSS)", Section 4. The RSA key sizes supported are: 3072 bits, and 4096-bit.
- Elliptical curve digital signature algorithm conforming to FIPS Pub 186-4, "Digital Signature Standard (DSS)", Section 5. The TOE supports curves P-256, P-384, and P-521.

Please refer to Table 14 for details.

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

PP Origin: *OSPP*

SFR Link: *FCS_COP.1(4)*

The TOE supports keyed-hash message authentication conforming to FIPS Pub 198-1 "The Keyed-Hash Message Authentication Code" and FIPS Pub 180-4 "Secure Hash Standard" with the following algorithms

- HMAC-SHA-256
- HMAC-SHA-384
- HMAC-SHA-512

The TOE supports key sizes 8 bits and higher for all HMAC algorithms.

Please refer to Table 14 for details.

7.2.2.8 FCS_RBG_EXT.1 Random Bit Generation

PP Origin: *OSPP*

SFR Link: *FCS_RBG_EXT.1*

The TOE uses a CTR_DRBG(AES) to generate random bits. The DRBG is seeded by an entropy source that accumulates entropy from a software-based noise (interrupts) source accessed via the kernel interface of getrandom. The entropy source provides seed data with a minimum of 256 bits of entropy to initially seed and reseed the DRBG.

Please refer to Table 14 for details.

7.2.2.9 FCS_STO_EXT.1 Storage of Sensitive Data

PP Origin: *OSPP*

SFR Link: *FCS_STO_EXT.1*

The TOE stores the following sensitive data.

- Usernames and passwords are used for authentication are maintained by local directory services in files protected by the operating system. The file with user names is write protected except for the root user. The credential store is read/write protected except for the root user.
- Trusted Certificates are used for establishing TLS, SSH sessions and are stored in files protected by the operating system. Per-user SSH keys are read-writable only for the owning user.
- Private Keys used for establishing TLS, SSH sessions and are stored in files protected by the operating system. Per-user SSH keys are read-writable only for the owning user.

The TOE offers a storage location in /etc/pki for private keys and certificates. User keys and certificates applicable to the SSH communication are stored in the user's home directory in the .ssh subdirectory. This subdirectory is only considered valid if its permission are set such that only the owner can access the files.

The TOE provides disk encryption for partitions. All partition including the root partition can be encrypted. In case of encryption of the root partition, the kernel and the initial RAM disk files must be stored on an unencrypted partition. These two files together with the boot loader configuration file cannot be stored protected by dm-crypt.

Confidentiality protected data storage

File system objects are stored on block devices, such as partitions on hard disk. The Linux operating systems offers the use of an additional layer between the file systems and the physical block device to encrypt and decrypt any data transmitted between the file system and the block device. The dm_crypt functionality uses the Linux device mapper to provide such encryption and decryption operation that is transparent to the file system and therefore to the user. The evaluated configuration uses AES-XTS as block cipher algorithm to protect data written to disk.

Before mounting the block device that is protected by the dm_crypt encryption scheme, the owner of the encrypted block device has to provide a passphrase. This passphrase is used to decrypt the symmetric volume key which is injected into the kernel. Using that volume key, the kernel is now able to decrypt (to unlock) the data on the device and provides access to data stored on that device. At this point, the file system can be mounted as the file system can now be read.

Once the dm_crypt protected device is unlocked and mounted, it is accessible as any other file system. When it is unmounted and locked (i.e. the kernel is informed to discard the volume key), all data on the device is inaccessible. Even administrative users like the root user are not able to access any data any more. When an administrator would access the raw hardware hosting the block device, only encrypted data can be read.

For the cryptographic operation, the creator of the dm_crypt block device can select the cipher. When creating the dm_crypt block device, the volume key is obtained from the a DRNG that is seeded by the Linux random number generator and stored on the block device encrypted with the user's passphrase. The key derivation mechanism from the user's password is based on the LUKS mechanism.

The encryption and decryption operation of the device is implemented by the kernel. To unlock the encrypted volume key stored on the protected block device, the cryptsetup application performs the following steps:

1. obtain the user's passphrase
2. apply the LUKS key derivation mechanism on the passphrase - the protection of the volume key is implemented with a Password-Based Key Derivation Function (PBKDF) according to the NIST documentation SP800-132 following option 2a outlined in section 5.4 of SP800-132 with the iteration count (specific to the used system as a calibration loop is used to run at least 100ms) and the used salt (generated by same RNG as used for generating the master volume key) stored in the LUKS header for the given volume key
3. read the encrypted volume key from the device
4. decrypt the volume key with the key derived from the user's passphrase
5. inject the decrypted volume key into the kernel and set up the mapping between the block device and the volume key

Using the cryptsetup tool, the volume key can also be transferred by encrypting it with another passphrase which can be given to another user. The transfer follows the same steps outlined for the unlocking operation, but instead of injecting the decrypted volume key into the kernel, cryptsetup fetches the new passphrase from the user, applies the LUKS mechanism on that passphrase, encrypts the volume key with the derived key and stores the encrypted volume key in a separate area on the device. At this point, the volume key is now stored encrypted in two separate places.

Similarly, the cryptsetup tool can be used to erase the storage location of one encrypted volume key which implies that the user owning the passphrase of the affected encrypted volume key is not able to unlock the block device any more.

The random number generator used to generate the key material is specified with FCS_RBG_EXT.1.

The installer allows the configuration of the full disk encryption schema where the entire disk is protected, except the /boot partition.

7.2.2.10 FCS_TLSC_EXT.1 TLS Client Protocol

PP Origin: *OSPP*

SFR Link: *FCS_TLSC_EXT.1*

The TOE provides TLSv1.2 to allow users from a remote host to establish a secure channel to the TOE. The TLS protocol performs the support authentication as part by verifying the RSA certificates. The TOE can be configured using a bi-directional certificate verification where the server side (implemented by the libvirt daemon) validates the client certificate.

The following table documents implementation details concerning the OpenSSL implementation's compliance to the relevant standards. It addresses areas where the standards permit different implementation choices such as optional features.

Reference	Description	Implementation Details
RFC 5246 section 7.3	Handshake protocol overview: certificates	The evaluated configuration always uses server certificates. Client certificates are used to allow the server to authenticate the client.
RFC 5246 appendix D.1	Random Number Generation and Seeding	OpenSSL uses data from the Linux kernel random number generator, a persistent entropy pool file, and volatile system statistics to seed the PRNG.
RFC 5246 appendix D.2	Certificates and authentication	The evaluated configuration supports verification of certificate chains.
RFC 5246 appendix A.5, RFC5288 section 3, RFC5289 sections 3 and 4	Cipher suites	The ciphers supported in the evaluated configuration are listed in FCS_TLSC_EXT.1 for the TLS protocol.
RFC 5246 appendix E	SSLv2, SSLv3, TLSv1.0, TLSv1.1 Backward Combatibility	The OpenSSL implementation supports the backwards compatible protocol, but this is disabled in the evaluated configuration. It permits use of TLSv1.2 exclusively.

Table 15: TLS implementation notes

The TOE supports the generation of the RSA key pair used by the server. The key generation mechanism uses the Linux kernel random number generator as entropy source. The evaluated configuration also allows the use of an externally-generated certificate. A widely accepted Certification Authority might be used to generate and/or sign such a certificate (allowing a wide community trusting this CA to validate the certificate). In a closed community it might also be sufficient to have one server within the community to act as a CA. The OpenSSL library provides the functions to set up such a CA, but those functions are not subject of this Security Target.

The key material used by TLS and SSH is obtained from the getrandom system call and is fed into the deterministic random number generator instantiated for the process performing the particular network communication.

The TOE implements TLS 1.2 (RFC 5246) client functionality 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

The TOE also supports the following TLS 1.2 cipher suites not specified in [OSPP].

- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA

The TOE verifies that server certificate is valid according to FIA_X509_EXT.1 and that the presented identifier matches the reference identifier according to RFC 6125. The reference identifiers supported are DNS and IP addresses in the SAN. Wildcards are supported. The TOE does not support certificate pinning. The TOE does not establish a trusted channel if the server certificate is invalid.

The TOE supports mutual authentication using X.509v3 certificates. When configured, the TOE will send a Certificate and Certificate Verify message in response to a Certificate Request message from a TLS server. It also implements session renegotiation.

7.2.2.11 FCS_TLSC_EXT.2 TLS Client Support for Supported Groups Extension

PP Origin: *OSPP*

SFR Link: *FCS_TLSC_EXT.2*

The TOE, by default, presents the Supported Elliptic Curves Extension in the Client Hello with the following NIST curves: secp256r1, secp384r1, and secp521r1. In addition, the safe primes from RFC7919 are supported as well.

7.2.2.12 FCS_TLSC_EXT.4 TLS Client Support for Mutual Authentication

PP Origin: *OSPP*

SFR Link: *FCS_TLSC_EXT.4*

The TOE supports mutual authentication using X.509v3 certificates. When configured, the TOE will send a Certificate and Certificate Verify message in response to a Certificate Request message from a TLS server.

7.2.2.13 FCS_SSH_EXT.1 SSH Protocol

PP Origin: *SSH*

SFR Link: *FCS_SSH_EXT.1*

The TOE provides the Secure Shell Protocol Version 2 (SSH v2.0) to allow users from a remote host to establish a secure connection and perform a logon to the TOE.

The following table documents implementation details concerning the OpenSSH implementation’s compliance to the relevant standards. It addresses areas where the standards permit different implementation choices such as optional features.

Reference	Description	Implementation Details
RFC 4253 chapter 5	Compatibility with old SSH versions	The OpenSSH implementation is capable of interoperating with clients and servers using the old 1.x protocol. That functionality is explicitly disabled in the evaluated configuration, it permits protocol version 2.0 exclusively.
RFC 4253 section 6.2	Compression	OpenSSH supports the OPTIONAL "zlib" compression method.
RFC 4253 section 6.3	Encryption	The ciphers supported in the evaluated configuration are listed in FCS_SSH_EXT.1 for the SSH protocol.
RFC 4252 chapter 7	Public Key Authentication Method: "publickey"	This REQUIRED authentication method is supported by OpenSSH but can be disabled by the administrator of the OpenSSH daemon.
RFC 4252 chapter 8	Password Authentication Method: "password"	This SHOULD authentication method is supported by OpenSSH but can be disabled by the administrator of the OpenSSH daemon.
RFC 4252 chapter 8	Password change request and setting new password	The OpenSSH implementation supports the optional password change mechanism in the evaluated configuration.
RFC 4252 chapter 9	Host-Based Authentication: "hostbased"	This OPTIONAL authentication method is disabled in the evaluated configuration.

Table 16: SSH implementation notes

The TOE supports the generation of RSA, DSA (for DH operation) as well as ECDSA key pairs. These key pairs are used by OpenSSH for the host keys as well as for the per-user keys. When a user registers his public key with the user he wants to access on the server side, a key-based authentication can be performed instead of a password-based authentication. The key generation mechanism uses the random number generator of the underlying cryptographic library. The evaluated configuration permits the import of externally-generated key pairs.

The TOE supports the following security functions of the SSH v2.0 protocol:

- Establishing a secure communication channel using the following cryptographic functions provided by the SSH v2.0 protocol:
  - Encryption as defined in section 6.3 of [RFC4253] - the DH/ECDH forming the basis for the key agreement and thus the symmetric / MAC keys are generated using the random number generator of the underlying cryptographic library;
  - Diffie-Hellman key agreement used in conjunction with the key derivation function as defined in section 7.2 of [RFC4253] supplemented by [RFC5656] chapter 4;
  - The keyed hash function for integrity protection as defined in section 6.4 of [RFC4253].

  Note: The protocol supports more cryptographic algorithms than the ones listed above. Those other algorithms are not covered by this evaluation and should be disabled or not used when running the evaluated configuration.
- Performing user authentication using the standard password-based authentication method the TOE provides for users (password authentication method as defined in chapter 5 of [RFC4252]).
- Performing user authentication using a RSA, or ECDSA key-based authentication method (public key authentication method as defined in chapter 5 of [RFC4252]).

- Checking the integrity of the messages exchanged and close down the connection in case an integrity error is detected.

The OpenSSH applications of sshd, ssh and ssh-keygen use the OpenSSL random number generator seeded by the getrandom system call to generate cryptographic keys. OpenSSL provides different DRNGs depending whether the FIPS 140-2 mode is enabled in the system.

The cryptographic implementations ensure that sensitive data is appropriately zeroized before releasing the associated memory.

The TOE supports the following authentication mechanisms with SSH:

- Password-based authentication
- Key-based authentication with RSA keys and ECDSA keys.

The TOE maintains a counter for each SSH packet which is increased by the number of received bytes. If the counter reaches the threshold of 262144 bytes, the connection is closed.

After processing at most 2^30 bytes covering both sent and received data or the last re-key is more than 1 hour ago, the TOE initiates a re-keying, when the option RekeyLimit is set appropriately.

The SSH implementation supports the following ciphers:

- Symmetric ciphers: aes128-cbc, aes256-cbc, aes128-gcm@openssh.com, aes256-gcm@openssh.com
- Asymmetric ciphers: rsa-sha2-256, rsa-sha2-512, ecdsa-sha2-nistp256, ecdsa-sha2-nistp384
- MAC: hmac-sha2-256, hmac-sha2-512, AES-GCM
- Key agreement: diffie-hellman-group16-sha512, diffie-hellman-group18-sha512, ecdh-sha2-nistp256, ecdh-sha2-nistp384, ecdh-sha2-nistp521

The aforementioned statements apply equally to the SSH client and server implementation.

7.2.2.14 FCS_SSHC_EXT.1 SSH Protocol - Client

PP Origin: *SSH*

SFR Link: *FCS_SSHC_EXT.1*

The TOE provides an SSH client which supports the SSH server authentication using RSA, and ECDSA.

7.2.2.15 FCS_SSHS_EXT.1 SSH Protocol - Server

PP Origin: *SSH*

SFR Link: *FCS_SSHS_EXT.1*

The TOE provides an SSH server which supports the SSH client authentication using RSA, and ECDSA.

7.2.3 User data protection

7.2.3.1 FDP_ACF_EXT.1 Access Controls for Protecting User Data

PP Origin: *OSPP*

SFR Link: *FDP_ACF_EXT.1*

The TOE uses the BtrFS file system which provides access control to data. File system object attributes includes manipulation of metadata (e.g., change, access, modify time), as well as owner and permission data (e.g., group-ids for allowing multiple users to have the same access

privileges, user-ids for individual access privileges, and permissions that can be assigned per user or group). The TOE provides the following file system security schemes: SELinux access control, POSIX access control lists (ACLs), Unix (BSD) permissions.

These security schemes fit together as follows (rules are processed in the following order).

1. If the SELinux rules forbids the requested access, the request is denied.
2. If an access control entry exists on the file, it is evaluated and used to determine access rights.
3. Otherwise, if the user ID matches the owner of the file, the "user" permissions (also called "owner" permissions) are used.
4. Otherwise, if the group ID matches the group for the file, the "group" permissions are used.
5. Otherwise, the "other" permissions are used.

SELinux

The TOE supports the use of SELinux to limit an app’s ability to access files. These limits override any permissions the app might otherwise have. SELinux rules are subtractive, not additive. Therefore, the file system permissions represent the maximum access an app might be allowed if SELinux also permits that access.

POSIX ACLs

The TOE supports ACLs, which are data structures that provide much more detailed control over permissions than Unix permissions. For example, ACLs allow the system administrator to specify that a specific user can delete a file but cannot write to it. An ACL consists of an ordered list of ACEs (access control entries), each of which associates a user or group with a set of permissions and specifies whether each permission is allowed or denied. ACEs also include attributes related to inheritance.

Each ACL on a directory can contain any combination of the following inheritance flags.

- Inherited (this ACE was inherited)
- File Inherit (this ACE should be inherited by files created within this directory)
- Directory Inherit (this ACE should be inherited by directories created within this directory)
- Inherit Only (this ACE should not be checked during authorization)
- No Propagate Inherit (this ACE should be inherited only by direct children; that is, the ACE should lose any Directory Inherit or File Inherit bit when inherited)

When it creates a new file, the kernel goes through the entire access control list of the parent directory and copies to the file's ACL any ACEs that are marked for file inheritance. Similarly, when it creates a new subdirectory, the kernel copies to the subdirectory's ACL any ACEs that are marked for directory inheritance.

If a file is copied and pasted into a directory, the kernel replicates the contents of the source file into a new file at the destination. Because it is creating a new file, the system checks the ACL of the parent directory and adds any inherited ACEs to whatever ACEs were in the original file. If a file is moved into a directory, on the other hand, the original file is not replicated and no ACEs are inherited. In this case, the parent directory's ACEs are added to the moved file only if the administrator specifically propagates ACEs from the parent directory through contained files and subdirectories. Similarly, once a file has been created, changing the ACL of the parent directory does not affect the ACL of contained files and subdirectories unless the administrator specifically propagates the change.

The order in which ACEs are placed in an ACL—and therefore the order in which they are evaluated to determine permissions—is as follows:

1. Explicitly specified deny associations

2. Explicitly specified allow associations

Inherited associations, in the same order in which they appeared in the parent. Since ACEs can be inherited, administrators can control the fine-grained permissions of files created in a directory by assigning inheritable ACEs to the directory. Doing so saves the work of assigning ACEs to each file individually. In addition, because ACEs can apply to groups of users, administrators can assign permissions to groups rather than having to specify permissions for each individual. Applying access security to directories and groups rather than to files and individuals saves administrator time and gives better file system performance in many circumstances.

Unix Permissions

Each file system object has a set of UNIX permissions defined by three attributes

- UID, short for user ID. Commonly referred to as the File’s Owner.
- GID, short for group ID.
- Flags that include permission bits and other related attributes.

The flags for a file or directory are a 16-bit value that is often represented as a three-digit or four-digit octal value (with the top four or seven bits dropped). The Owner, Group, and Other bit sets contain three bits: read, write, execute (rwx for short).

7.2.4 Identification and authentication

7.2.4.1 FIA_AFL.1 Authentication failure handling (Refined)

PP Origin: *OSPP*

SFR Link: *FIA_AFL.1*

The TOE will detect when an administrator configurable integer within 1 and 2^32 - 1 unsuccessful authentication attempts for authentication based on username and password attempts as well as SSH-based authentication attempts have been met. Once the specified number of unsuccessful authentication attempts for an account has been met, the TOE will lock out the account.

7.2.4.2 FIA_UAU.5 Multiple Authentication Mechanisms (Refined)

PP Origin: *OSPP*

SFR Link: *FIA_UAU.5*

The TOE supports authentication based on username/password.

For password-based authentication, the user account contains a username and a password. A random salt is created for the password which is used to derive a SHA2-512 hash value that is stored in the /etc/shadow file. When a user logs into the system, the TOE uses the entered password and the randomly generated salt and compares this with the stored value. If they match, then the user is granted access to the system. If the values do not match, then the user is not granted access.

The TOE supports authentication with SSH-keys. The public key is stored with the SSH server that a user wants to use for the key-based authentication. Similarly, an SSH client authenticates the remote SSH server's public key against his local database of public keys. SSH-key-based authentication is specified in RFC4252.

7.2.4.3 FIA_X509_EXT.1 X.509 Certificate Validation

PP Origin: *OSPP*

SFR Link: *FIA_X509_EXT.1*

When an X.509 certificate is presented as part of the TLS connection establishment, the TOE verifies the certificate path, and certification validation process by verifying the following rules

- RFC 5280 certificate validation and certificate path validation
- The certificate path must terminate with a trusted CA certificate marked as a trust anchor in the user or platform’s keychain
- All CA certificates contain the basicConstraints extension with the CA flag is set to TRUE and (if present) path constraints are met
- All CA certificate includes caSigning purpose in the key usage field
- Certificate revocation status is checked using OCSP stapling as well as CRL. The certificate is accepted if it's revocation status cannot be determined.

The TOE validates the extendedKeyUsage field depending on the specific usage of the certificate as follows.

- 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)

X.509 certificates are validated when imported into the TOE's trusted certificate store, during session establishment with a peer and prior to presenting a certificate to the peer during trusted channel implementation using TLS for mutual authentications.

The TOE implements X.509 certificate chain validation following RFC5280 and OCSP following RFC6066 and CRL as specified in RFC8603.

7.2.4.4 FIA_X509_EXT.2 X.509 Certificate Authentication

PP Origin: *OSPP*

SFR Link: *FIA_X509_EXT.2*

The TOE uses X.509v3 certificates for performing mutual authentication for TLS in HTTPS connections.

7.2.5 Security management

7.2.5.1 FMT_MOF_EXT.1 Management of security functions behavior

PP Origin: *OSPP*

SFR Link: *FMT_MOF_EXT.1*

See the TSS for FMT_SMF_EXT.1.

7.2.5.2 FMT_SMF_EXT.1 Specification of Management Functions

PP Origin: *OSPP*

SFR Link: *FMT_SMF_EXT.1*

The TOE supports the following roles: Administrator and User. The Administrator is a member of the local admin group or an applied configuration profile, and the User is an unprivileged account.

The Administrator has access to the following management functions.

- Enable/disable screen lock
- Configure screen lock inactivity timeout
- Import keys/secrets into the secure key storage (for the system)
- 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 limiting number of attempts during a time period
- Configure host-based firewall
- Configure name/address of the logging server (syslog) to which to send logging records
- Configure audit rules
- Configure name/address of network time server
- Enable/disable automatic software update
- Configure Wi-Fi interface

Access to the administrator-only management functions is restricted by means of appropriate permission bits applied to the configuration files that hold the respective configuration data.

The User has access to the following management functions.

- Enable/disable screen lock
- Configure screen lock inactivity timeout when locked through the screen saver
- Import keys/secrets into the secure key storage (for the user)
- Configure Wi-Fi interface (if not restricted by an Administrator)

7.2.6 Protection of the TSF

7.2.6.1 FPT_ACF_EXT.1 Access controls

PP Origin: *OSPP*

SFR Link: *FPT_ACF_EXT.1*

The TOE provides access control policy through file permissions protecting system files preventing users from modifying protected files and folders. Appropriate permission settings protect the following parts of the system from unauthorized modification.

- Kernel / initial RAM disk / boot loader configuration:
  - /boot
- Kernel drivers and modules
  - /lib/modules
- Shared libraries
  - /usr/lib
  - /usr/lib64

    - /lib
    - /lib64
- Applications
    - /bin
    - /sbin
    - /usr/sbin
    - /usr/bin
- Security audit logs
    - /var/log/audit
- System configuration files
    - /etc
- TSF-data
    - /var except /var/tmp which is a link to /tmp
- System-wide credentials repositories
    - /etc/group (world-readable)
    - /etc/passwd (world-readable)
    - /etc/gshadow
    - /etc/shadow
    - /etc/security/opasswd

The TOE prevents unprivileged users from reading Security audit logs. System-wide credential repositories are write-protected for unprivileged users. In addition, the repository holding the user passwords is read/write protected for unprivileged users.

7.2.6.2 FPT_ASLR_EXT.1 Address Space Layout Randomization

PP Origin: *OSPP*

SFR Link: *FPT_ASLR_EXT.1*

The TOE always randomizes process address memory locations with 11 bits (stack) and 28 bits (text segment) of entropy.

7.2.6.3 FPT_SBOP_EXT.1 Stack Buffer Overflow Protection

PP Origin: *OSPP*

SFR Link: *FPT_SBOP_EXT.1*

The TOE protects all TOE binaries from stack-based buffer overflow attacks using.

- ASLR to randomize the locations of the stack, preventing attackers from jumping to specific data that has been written to the stack.
- Stack canaries to detect if the stack has been overwritten when returning from a function.

The following list enumerates the binaries which are exempt from the stack buffer overflow protection. The protection mechanisms could not be applied due to technical reasons including the fact that the "stack-protector-strong" compiler option was used. This option protects function calling "alloca", and functions with buffers larger than or equal to 8 bytes.

- /lib/modules/5.14.21-150400.*-default/vdso/vdso*.so
- /usr/lib64/libstdc++.so.6.*
- /lib64/ld-2.31.so
- /lib64/libgcc_s.so.1

- /lib64/libmvec-2.31.so
- /lib64/libtinfow.so.6.1
- /lib64/multipath/libprioalua.so
- /lib64/multipath/libprioconst.so
- /lib64/security/pam_cockpit_cert.so
- /lib64/security/pam_deny.so
- /sbin/blogger
- /sbin/ttyrun
- /usr/bin/ausyscall
- /usr/bin/bzip2recover
- /usr/bin/clear
- /usr/bin/conmon
- /usr/bin/dbus-uuidgen
- /usr/bin/grub2-editenv
- /usr/bin/grub2-emu
- /usr/bin/grub2-file
- /usr/bin/grub2-fstest
- /usr/bin/grub2-glue-efi
- /usr/bin/grub2-menulst2cfg
- /usr/bin/grub2-mkfont
- /usr/bin/grub2-mkimage
- /usr/bin/grub2-mklayout
- /usr/bin/grub2-mknetdir
- /usr/bin/grub2-mkpasswd-pbkdf2
- /usr/bin/grub2-mkrelpath
- /usr/bin/grub2-mkrescue
- /usr/bin/grub2-mkstandalone
- /usr/bin/grub2-mount
- /usr/bin/grub2-protect
- /usr/bin/grub2-render-label
- /usr/bin/grub2-script-check
- /usr/bin/grub2-syslinux2cfg
- /usr/bin/ignition-validate
- /usr/bin/line
- /usr/bin/mandb
- /usr/bin/podman
- /usr/bin/python3.6
- /usr/bin/python3.6m
- /usr/bin/rev
- /usr/bin/semodule_unpackage
- /usr/bin/suseconnect
- /usr/bin/tabs
- /usr/bin/xxd
- /usr/lib/perl5/5.26.1/aarch64-linux-thread-multi/auto/Devel/Peek/Peek.so

- /usr/lib/perl5/5.26.1/aarch64-linux-thread-multi/auto/Encode/Byte/Byte.so
- /usr/lib/perl5/5.26.1/aarch64-linux-thread-multi/auto/Encode/CN/CN.so
- /usr/lib/perl5/5.26.1/aarch64-linux-thread-multi/auto/Encode/EBCDIC/EBCDIC.so
- /usr/lib/perl5/5.26.1/aarch64-linux-thread-multi/auto/Encode/JP/JP.so
- /usr/lib/perl5/5.26.1/aarch64-linux-thread-multi/auto/Encode/KR/KR.so
- /usr/lib/perl5/5.26.1/aarch64-linux-thread-multi/auto/Encode/Symbol/Symbol.so
- /usr/lib/perl5/5.26.1/aarch64-linux-thread-multi/auto/Encode/TW/TW.so
- /usr/lib/perl5/5.26.1/aarch64-linux-thread-multi/auto/Fcntl/Fcntl.so
- /usr/lib/perl5/5.26.1/aarch64-linux-thread-multi/auto/File/DosGlob/DosGlob.so
- /usr/lib/perl5/5.26.1/aarch64-linux-thread-multi/auto/Filter/Util/Call/Call.so
- /usr/lib/perl5/5.26.1/aarch64-linux-thread-multi/auto/I18N/Langinfo/Langinfo.so
- /usr/lib/perl5/5.26.1/s390x-linux-thread-multi/auto/Devel/Peek/Peek.so
- /usr/lib/perl5/5.26.1/s390x-linux-thread-multi/auto/Encode/Byte/Byte.so
- /usr/lib/perl5/5.26.1/s390x-linux-thread-multi/auto/Encode/CN/CN.so
- /usr/lib/perl5/5.26.1/s390x-linux-thread-multi/auto/Encode/EBCDIC/EBCDIC.so
- /usr/lib/perl5/5.26.1/s390x-linux-thread-multi/auto/Encode/JP/JP.so
- /usr/lib/perl5/5.26.1/s390x-linux-thread-multi/auto/Encode/KR/KR.so
- /usr/lib/perl5/5.26.1/s390x-linux-thread-multi/auto/Encode/Symbol/Symbol.so
- /usr/lib/perl5/5.26.1/s390x-linux-thread-multi/auto/Encode/TW/TW.so
- /usr/lib/perl5/5.26.1/s390x-linux-thread-multi/auto/Fcntl/Fcntl.so
- /usr/lib/perl5/5.26.1/s390x-linux-thread-multi/auto/File/DosGlob/DosGlob.so
- /usr/lib/perl5/5.26.1/s390x-linux-thread-multi/auto/Filter/Util/Call/Call.so
- /usr/lib/perl5/5.26.1/s390x-linux-thread-multi/auto/I18N/Langinfo/Langinfo.so
- /usr/lib/perl5/5.26.1/x86_64-linux-thread-multi/auto/Devel/Peek/Peek.so
- /usr/lib/perl5/5.26.1/x86_64-linux-thread-multi/auto/Encode/Byte/Byte.so
- /usr/lib/perl5/5.26.1/x86_64-linux-thread-multi/auto/Encode/CN/CN.so
- /usr/lib/perl5/5.26.1/x86_64-linux-thread-multi/auto/Encode/EBCDIC/EBCDIC.so
- /usr/lib/perl5/5.26.1/x86_64-linux-thread-multi/auto/Encode/JP/JP.so
- /usr/lib/perl5/5.26.1/x86_64-linux-thread-multi/auto/Encode/KR/KR.so
- /usr/lib/perl5/5.26.1/x86_64-linux-thread-multi/auto/Encode/Symbol/Symbol.so
- /usr/lib/perl5/5.26.1/x86_64-linux-thread-multi/auto/Encode/TW/TW.so
- /usr/lib/perl5/5.26.1/x86_64-linux-thread-multi/auto/Fcntl/Fcntl.so
- /usr/lib/perl5/5.26.1/x86_64-linux-thread-multi/auto/File/DosGlob/DosGlob.so
- /usr/lib/perl5/5.26.1/x86_64-linux-thread-multi/auto/Filter/Util/Call/Call.so
- /usr/lib/perl5/5.26.1/x86_64-linux-thread-multi/auto/I18N/Langinfo/Langinfo.so
- /usr/lib/perl5/vendor_perl/5.26.1/aarch64-linux-thread-multi/auto/Locale/gettext_xs/gettext_xs.so
- /usr/lib/perl5/vendor_perl/5.26.1/s390x-linux-thread-multi/auto/Locale/gettext_xs/gettext_xs.so
- /usr/lib/perl5/vendor_perl/5.26.1/x86_64-linux-thread-multi/auto/Locale/gettext_xs/gettext_xs.so
- /usr/lib64/ModemManager/libmm-plugin-broadmobi.so
- /usr/lib64/ModemManager/libmm-plugin-dlink.so
- /usr/lib64/ModemManager/libmm-plugin-fibocom.so
- /usr/lib64/ModemManager/libmm-plugin-foxconn.so
- /usr/lib64/ModemManager/libmm-plugin-generic.so
- /usr/lib64/ModemManager/libmm-plugin-gosuncn.so

- /usr/lib64/ModemManager/libmm-plugin-haier.so
- /usr/lib64/ModemManager/libmm-plugin-nokia-icera.so
- /usr/lib64/ModemManager/libmm-plugin-novatel.so
- /usr/lib64/ModemManager/libmm-plugin-option.so
- /usr/lib64/ModemManager/libmm-plugin-qcom-soc.so
- /usr/lib64/ModemManager/libmm-plugin-samsung.so
- /usr/lib64/ModemManager/libmm-plugin-sierra.so
- /usr/lib64/ModemManager/libmm-plugin-telit.so
- /usr/lib64/ModemManager/libmm-plugin-tplink.so
- /usr/lib64/engines-1.1/capi.so
- /usr/lib64/engines-1.1/padlock.so
- /usr/lib64/gawk/readdir.so
- /usr/lib64/gawk/revtwoway.so
- /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
- /usr/lib64/ipsec/plugins/libstrongswan-aes.so
- /usr/lib64/ipsec/plugins/libstrongswan-mgf1.so
- /usr/lib64/ipsec/plugins/libstrongswan-nonce.so
- /usr/lib64/ipsec/plugins/libstrongswan-random.so
- /usr/lib64/libbd_part_err.so.2.0.0
- /usr/lib64/libbrotlicommon.so.1.0.7
- /usr/lib64/libicudata.so.suse65.1
- /usr/lib64/liblttng-ust-fd.so.0.0.0
- /usr/lib64/libpanel.so.6.1
- /usr/lib64/libpanelw.so.6.1
- /usr/lib64/libplc4.so
- /usr/lib64/liburcu-common.so.6.1.0
- /usr/lib64/liburing.so.2.1.0
- /usr/lib64/libx86emu.so.3.1
- /usr/lib64/libxentoolcore.so.1.0
- /usr/lib64/python3.6/site-packages/_dbus_glib_bindings.so
- /usr/lib64/python3.6/site-packages/markupsafe/_speedups.cpython-36m-aarch64-linux-gnu.so
- /usr/lib64/python3.6/site-packages/markupsafe/_speedups.cpython-36m-s390x-linux-gnu.so
- /usr/lib64/python3.6/site-packages/markupsafe/_speedups.cpython-36m-x86_64-linux-gnu.so
- /usr/lib64/rpm-plugins/prioreset.so
- /usr/lib64/rpm-plugins/syslog.so
- /usr/lib64/xenfsimage/iso9660/fsimage.so
- /usr/lib64/xenfsimage/ufs/fsimage.so
- /usr/lib64/xtables/libebt_arpreply.so
- /usr/lib64/xtables/libebt_dnat.so

- /usr/lib64/xtables/libebt_redirect.so
- /usr/lib64/xtables/libebt_snat.so
- /usr/lib64/xtables/libip6t_DNPT.so
- /usr/lib64/xtables/libip6t_HL.so
- /usr/lib64/xtables/libip6t_LOG.so
- /usr/lib64/xtables/libip6t_REJECT.so
- /usr/lib64/xtables/libip6t_SNAT.so
- /usr/lib64/xtables/libip6t_SNPT.so
- /usr/lib64/xtables/libip6t_ah.so
- /usr/lib64/xtables/libip6t_eui64.so
- /usr/lib64/xtables/libip6t_frag.so
- /usr/lib64/xtables/libip6t_hl.so
- /usr/lib64/xtables/libip6t_ipv6header.so
- /usr/lib64/xtables/libip6t_rt.so
- /usr/lib64/xtables/libip6t_srh.so
- /usr/lib64/xtables/libipt_CLUSTERIP.so
- /usr/lib64/xtables/libipt_ECN.so
- /usr/lib64/xtables/libipt_LOG.so
- /usr/lib64/xtables/libipt_REJECT.so
- /usr/lib64/xtables/libipt_TTL.so
- /usr/lib64/xtables/libipt_ULOG.so
- /usr/lib64/xtables/libipt_ah.so
- /usr/lib64/xtables/libipt_ttl.so
- /usr/lib64/xtables/libxt_AUDIT.so
- /usr/lib64/xtables/libxt_CHECKSUM.so
- /usr/lib64/xtables/libxt_CONNMARK.so
- /usr/lib64/xtables/libxt_CONNSECMARK.so
- /usr/lib64/xtables/libxt_DSCP.so
- /usr/lib64/xtables/libxt_HMARK.so
- /usr/lib64/xtables/libxt_IDLETIMER.so
- /usr/lib64/xtables/libxt_LED.so
- /usr/lib64/xtables/libxt_NFLOG.so
- /usr/lib64/xtables/libxt_NFQUEUE.so
- /usr/lib64/xtables/libxt_SECMARK.so
- /usr/lib64/xtables/libxt_SYNPROXY.so
- /usr/lib64/xtables/libxt_TCPMSS.so
- /usr/lib64/xtables/libxt_TEE.so
- /usr/lib64/xtables/libxt_TOS.so
- /usr/lib64/xtables/libxt_TPROXY.so
- /usr/lib64/xtables/libxt_TRACE.so
- /usr/lib64/xtables/libxt_addrtype.so
- /usr/lib64/xtables/libxt_cgroup.so
- /usr/lib64/xtables/libxt_cluster.so
- /usr/lib64/xtables/libxt_connbytes.so

- /usr/lib64/xtables/libxt_connlimit.so
- /usr/lib64/xtables/libxt_connmark.so
- /usr/lib64/xtables/libxt_cpu.so
- /usr/lib64/xtables/libxt_dccp.so
- /usr/lib64/xtables/libxt_dscp.so
- /usr/lib64/xtables/libxt_ecn.so
- /usr/lib64/xtables/libxt_esp.so
- /usr/lib64/xtables/libxt_helper.so
- /usr/lib64/xtables/libxt_ipcomp.so
- /usr/lib64/xtables/libxt_ipvs.so
- /usr/lib64/xtables/libxt_length.so
- /usr/lib64/xtables/libxt_mac.so
- /usr/lib64/xtables/libxt_mark.so
- /usr/lib64/xtables/libxt_multiport.so
- /usr/lib64/xtables/libxt_nfacct.so
- /usr/lib64/xtables/libxt_osf.so
- /usr/lib64/xtables/libxt_physdev.so
- /usr/lib64/xtables/libxt_pkttype.so
- /usr/lib64/xtables/libxt_policy.so
- /usr/lib64/xtables/libxt_quota.so
- /usr/lib64/xtables/libxt_recent.so
- /usr/lib64/xtables/libxt_rpfilter.so
- /usr/lib64/xtables/libxt_sctp.so
- /usr/lib64/xtables/libxt_socket.so
- /usr/lib64/xtables/libxt_standard.so
- /usr/lib64/xtables/libxt_statistic.so
- /usr/lib64/xtables/libxt_tcpmss.so
- /usr/lib64/xtables/libxt_tos.so
- /usr/lib64/xtables/libxt_udp.so
- /usr/sbin/accessdb
- /usr/sbin/cnitool
- /usr/sbin/compute_av
- /usr/sbin/compute_create
- /usr/sbin/compute_member
- /usr/sbin/compute_relabel
- /usr/sbin/cracklib-unpacker
- /usr/sbin/ebtables-legacy
- /usr/sbin/ebtablesu
- /usr/sbin/fatresize
- /usr/sbin/findfs
- /usr/sbin/getenforce
- /usr/sbin/getfilecon
- /usr/sbin/getpidcon
- /usr/sbin/getseuser

- /usr/sbin/grub2-bios-setup
- /usr/sbin/grub2-install
- /usr/sbin/grub2-macbless
- /usr/sbin/grub2-ofpathname
- /usr/sbin/grub2-probe
- /usr/sbin/grub2-sparc64-setup
- /usr/sbin/ipset
- /usr/sbin/load_policy
- /usr/sbin/mdevctl
- /usr/sbin/mklost+found
- /usr/sbin/open_init_pty
- /usr/sbin/partprobe
- /usr/sbin/pivot_root
- /usr/sbin/policyvers
- /usr/sbin/runc
- /usr/sbin/selinux_check_access
- /usr/sbin/selinux_check_securetty_context
- /usr/sbin/selinuxenabled
- /usr/sbin/semodule
- /usr/sbin/setenforce
- /usr/sbin/setfilecon
- /usr/sbin/togglesebool
- /usr/sbin/validatetrans
- /usr/lib64/libabsl_bad_any_cast_impl.so.2308.0.0
- /usr/lib64/libabsl_bad_optional_access.so.2308.0.0
- /usr/lib64/libabsl_bad_variant_access.so.2308.0.0
- /usr/lib64/libabsl_city.so.2308.0.0
- /usr/lib64/libabsl_cordz_functions.so.2308.0.0
- /usr/lib64/libabsl_cordz_sample_token.so.2308.0.0
- /usr/lib64/libabsl_crc_cpu_detect.so.2308.0.0
- /usr/lib64/libabsl_die_if_null.so.2308.0.0
- /usr/lib64/libabsl_exponential_biased.so.2308.0.0
- /usr/lib64/libabsl_flags.so.2308.0.0
- /usr/lib64/libabsl_flags_commandlineflag.so.2308.0.0
- /usr/lib64/libabsl_flags_commandlineflag_internal.so.2308.0.0
- /usr/lib64/libabsl_hash.so.2308.0.0
- /usr/lib64/libabsl_leak_check.so.2308.0.0
- /usr/lib64/libabsl_log_entry.so.2308.0.0
- /usr/lib64/libabsl_log_initialize.so.2308.0.0
- /usr/lib64/libabsl_log_internal_conditions.so.2308.0.0
- /usr/lib64/libabsl_log_internal_globals.so.2308.0.0
- /usr/lib64/libabsl_log_internal_nullguard.so.2308.0.0
- /usr/lib64/libabsl_log_internal_proto.so.2308.0.0
- /usr/lib64/libabsl_log_severity.so.2308.0.0

- /usr/lib64/libabsl_log_sink.so.2308.0.0
- /usr/lib64/libabsl_low_level_hash.so.2308.0.0
- /usr/lib64/libabsl_periodic_sampler.so.2308.0.0
- /usr/lib64/libabsl_random_internal_platform.so.2308.0.0
- /usr/lib64/libabsl_random_internal_randen.so.2308.0.0
- /usr/lib64/libabsl_random_internal_randen_hwaes.so.2308.0.0
- /usr/lib64/libabsl_random_internal_randen_hwaes_impl.so.2308.0.0
- /usr/lib64/libabsl_random_seed_gen_exception.so.2308.0.0
- /usr/lib64/libabsl_raw_hash_set.so.2308.0.0
- /usr/lib64/libabsl_spinlock_wait.so.2308.0.0
- /usr/lib64/libabsl_stacktrace.so.2308.0.0
- /usr/lib64/libabsl_strings_internal.so.2308.0.0
- /usr/lib64/libabsl_throw_delegate.so.2308.0.0

7.2.6.4 FPT_TST_EXT.1 Boot Integrity

PP Origin: *OSPP*

SFR Link: *FPT_TST_EXT.1*

When a computer with Linux is turned on, the operating system is loaded into memory by a special program called a boot loader. A boot loader usually exists on the system's primary hard drive, or other media device, and has the sole responsibility of loading the Linux kernel with its required files or, in some cases, other operating systems, into memory. Each architecture capable of running Linux uses a different boot loader.

The init process provided with the systemd suite and is the ancestor of all userspace processes and is process ID 1. Its main functions are to start processes and daemons based on the contents of the /etc/systemd/system, /usr/lib/systemd/ and /lib/systemd directories, to reap child processes, and to react to certain signals, such as start or stop signals.

Boot loader operation

A boot loader is a program that resides in the starting sectors of a disk, that is, the Master Boot Record (MBR) of the hard disk. After testing the system during boot, the Basic Input-Output System (BIOS) transfers control to the MBR if the system is set to be booted from there. Then the program residing in MBR gets executed. This program is called the boot loader. Its duty is to transfer control to the operating system, which will then proceed with the boot process.

The boot process consists of the following steps when the CPU is powered on or reset:

1. The firmware performs any hardware initialization steps.
2. The BIOS searches for the boot loader to boot in an order predefined by the firmware setting. Once a valid device is found, the firmware copies the contents of its first sector containing the boot loader into RAM, and starts executing the code just copied.

Every boot loader performs the following general steps to initialize Linux:

1. Loading the kernel image it is configured to load (the actual way of configuring the boot loader is different for each boot loader implementation). The loading process ensures that the kernel image is loaded to a well-defined memory location.
2. Loading the initramfs image it is configured to load. Again, this image is loaded to a well-defined memory location.
3. The kernel is compiled such that the setup function will always be loaded into a well-known memory location. This allows the boot loader to jump to the setup function to transfer control to the kernel.

The system firmware supports the boot integrity as follows:

- x86: UEFI implements the starting of the boot loader. During the start, UEFI performs a signature verification of the first stage boot loader executable called "shim boot loader" that also holds the certificate for validating the Grub boot loader. That signature is part of the boot loader EFI file and is signed by Microsoft since the default UEFI certificate in every system is the Microsoft certificate. The purpose of the "shim boot loader" is to perform the integrity check of the Grub boot loader binary and the certificate used to verify the subsequent software images. In turn, the Grub boot loader verifies the integrity of the kernel binary file. The loaded binary has a PKCS #7 signature at the end of the file which is used by the secure boot process to perform signature validation.
- IBM Z: The loaded binaries have a signature associated with the file which is used by the secure boot process to perform signature validation.
- ARM: The loaded binaries have a signature associated with the file which is used by the secure boot process to perform signature validation.

Kernel boot process

The following initialization process is followed by the kernel. The details of the boot process are very different for each architecture. However, the following high-level steps are followed by each architecture.

Note, the kernel binary is compressed, except for a small code portion. That portion contains the setup code and the decompression routines in the kernel code to allow the kernel code to decompress itself.

The following steps are performed by the kernel after being loaded by the boot loader.

1. The setup function reinitializes the hardware devices in the computer and sets up the environment for the execution of the kernel program. The setup function initializes and configures hardware devices, such as the keyboard, video card, disk controller, and floating point unit.
2. The kernel is loaded into memory, and if its a compressed image, it is decompressed.
3. The kernel calls a second start-up (e.g. startup_32 on x86) function to set the execution environment for process 0.
4. The kernel initializes the memory management system.
5. The kernel sets the kernel mode stack for process 0.
6. The kernel initializes the provisional Page Tables and enables paging.
7. The kernel sets up the exception handlers.
8. start_kernel completes the kernel initialization by initializing Page Tables, Memory Handling Data Structures, the SLUB allocator, system date, and system time.

User space boot process

After the kernel is fully initialized, the user space is started up. There are the following two phases covering the boot process:

- initramfs: This state is intended to perform any initialization work to make the root file system available, such as loading kernel modules with special drivers needed to access the non-volatile storage holding the root file system.
- systemd This state initializes the entire user space by loading applications and daemons and performs any setup and configuration process necessary to get the system into the operational state.

initramfs

The following steps are performed to initialize the initramfs:

1. After the kernel is loaded and initialized, it locates the compressed initramfs image in memory.
2. The Linux kernel uncompresses the image.
3. The kernel performs a loopback mount of the uncompressed initramfs image to mount it as the root file system.
4. The kernel executes the /linuxrc or /sbin/init executable. This is a copy of systemd which executes out of the initramfs.
5. systemd does whatever it needs to do to for setting up the system to allow accessing the root file system based on the configuration.
6. After the systemd application terminates, the kernel unmounts the initramfs, and mounts the root file system pointed to by the “root” kernel command line parameter.

systemd

On every Unix system there is one process with the special process identifier 1. It is started by the kernel before all other processes and is the parent process for all those other processes that have nobody else to be child of. Due to that it can do a lot of stuff that other processes cannot do. And it is also responsible for some things that other processes are not responsible for, such as bringing up and maintaining userspace during boot.

systemd starts up and supervises the entire system (hence the name...). It is based around the notion of units. In systemd, a unit refers to a resource that is managed. Each resource is defined by a configuration file called a unit file. Example: a unit avahi.service is the unit file for the Avahi daemon. Units are categorized by the type of their resource. The suffix portion of the unit's file name is the type.

The generic boot sequence after the initramfs operation is finished is given in the following. The kernel has mounted the root file system which hosts /sbin/init – note that this init application is implemented with the systemd framework. This application is the driver of the user space boot process.

The kernel executes /sbin/init which finalizes the boot sequence implemented in the kernel.

1. Before executing the /sbin/init application, it resets the pid table to assign process ID one to the init process.
2. systemd is an event-driven system as described in systemd(8).
3. The boot process driven by systemd is purely based on events. If one event is observed, all tasks associated with that event are executed in parallel. The implemented boot sequence with all events is outlined in bootup(7). The boot process covers the following aspects:
    a. Mounts the /proc special file system.
    b. Mounts the /dev/pts special file system.
    c. Generate the /etc/nologin file early in the boot process.
    d. Saves and restores the system entropy tool for higher quality random number generation.
    e. Configures network interfaces.
    f. Starts the system logging daemons.
    g. Starts the sshd daemon.
    h. Starts the cron daemon.
    i. Probes hardware for setup and configuration.
    j. Enables the logind daemon for authentication.

7.2.6.5 FPT_TUD_EXT.1 Trusted Update

PP Origin: *OSPP*

SFR Link: *FPT_TUD_EXT.1*

The TOE allows the user to check for and install updates using the zypper application. Using zypper, updates can be manually queried, downloaded and installed as well as automatically be installed. When an update is initiated, the TOE downloads the update package and performs the RSA 4096-bit digital signature verification. If the verification is successful, the TOE installs the update. If the verification is unsuccessful, the TOE terminates the updates process.

7.2.6.6 FPT_TUD_EXT.2 Trusted Update for Application Software

PP Origin: *OSPP*

SFR Link: *FPT_TUD_EXT.2*

See the TSS for FPT_TUD_EXT.1.

7.2.7 TOE access

7.2.7.1 FTA_TAB.1 Default TOE access banners

PP Origin: *OSPP*

SFR Link: *FTA_TAB.1*

The TOE will display an advisory warning message regarding unauthorized use of the OS prior to establishing a user session.

7.2.8 Trusted path/channels

7.2.8.1 FTP_ITC_EXT.1 Trusted channel communication

PP Origin: *OSPP*

SFR Link: *FTP_ITC_EXT.1*

The TOE uses TLS as conforming to FCS_TLSC_EXT.1 to provide a trusted channel between itself and authorized IT entities. The update server and other authenticated TLS servers are authorized IT entities.

The TOE uses SSHv2 as conforming to FCS_SSH_EXT.1 to provide a trusted channel between itself and authorized IT entities.

7.2.8.2 FTP_TRP.1 Trusted Path

PP Origin: *OSPP*

SFR Link: *FTP_TRP.1*

The TOE provides a trusted path using the cryptographic network protocols specified in this ST between itself and local users that provides assured identification of its endpoints. This trusted path based on SSH is used to allow remote administrators to securely access the TOE for administration.

8 Abbreviations, Terminology, and References

8.1 Abbreviations

ACE
Access Control Entry

AES
Advanced Encryption Standard

app
Application

API
Application Programming Interface

ASLR
Address Space Layout Randomization

BSD
Berkeley Software Distribution

BSM
Basic Security Module

CA
Certificate Authority

CBC
Cipher Block Chaining

CC
Common Criteria

CCM
Counter with CBC-MAC

CEM
Common Evaluation Methodology

CIFS
Common Internet File System

CMC
Certificate Management over CMS

CMS
Cryptographic Message Syntax

CSP
Critical Security Parameters

CTR
Counter Mode Block Chaining

CVE
Common Vulnerabilities and Exposures

DAR
Data At Rest

DEK
Data Encryption Key

DEP
Data Execution Prevention

DNS
Domain Name System

DRBG
Deterministic Random Bit Generator

DSS
Digital Signature Standard

ECC
Elliptic Curve Cryptography

ECDH
Elliptic Curve Diffie-Hellman

ECDHE
ECDH Ephemeral

EKU
extendedKeyUsage

EST
Enrollment over Secure Transport

GCM
Galois/Counter Mode

GID
Group Identifier

GPOS
General Purpose Operating System

HCI
Host Controller Interface

HMAC
Keyed-hash Message Authentication Code

HTTPS
Hypertext Transfer Protocol Secure

ID
Identifier or Identity

IP
Internet Protocol

KAS
Key Agreement Scheme

KEK
Key Encryption Key

MAC
Message Authentication Code

OCSP
Online Certificate Status Protocol

OID
Object Identifier

OS
Operating System

PBKDF
Password-Based Key Derivation Function

PII
Personally Identifiable Information

PIN
Personal Identification Number

PKI
Public Key Infrastructure

POSIX
Portable Operating System Interface

PP
Protection Profile

RA
Registration Authority

RBG
Random Bit Generator

ROM
Read Only Memory

RSA
Rivest-Shamir-Adleman

SAN
Subject Alternative Name

SAR
Security Assurance Requirement

SCEP
Simple Certificate Enrollment Protocol

SEP
Secure Enclave Processor

SFR
Security Functional Requirement

SHA
Secure Hash Algorithm

SMB
Server Message Block

SoC
System on a Chip

ST
Security Target

TLS
Transport Layer Security

TOE
Target of Evaluation

TSF
TOE Security Functionality

TSFI
TSF Interface

TSS
TOE Summary Specification

UDID
Unique Device Identifier

UID
User Identifier

UUID
Universally Unique Identifier

XTS
XEX-based tweaked-codebook mode with ciphertext stealing

8.2 Terminology

This section contains definitions of technical terms that are used with a meaning specific to this document. Terms defined in the [CC] are not reiterated here, unless stated otherwise.

Administrator
An administrator is responsible for management activities, including setting policies that are applied by the enterprise on the operating system. This administrator could be acting remotely through a management server, from which the system receives configuration policies. An administrator can enforce settings on the system which cannot be overridden by non-administrator users.

API
A specification of routines, data structures, object classes, and variables that allows an application to make use of services provided by another software component, such as a library. APIs are often provided for a set of libraries included with the platform.

app
Software that runs on a platform and performs tasks on behalf of the user or owner of the platform, as well as its supporting documentation.

AppArmor
Linux kernel LSM module that is able to implement additional restrictions for executables. This LSM is unused in the evaluated configuration.

ASLR
An anti-exploitation feature which loads memory mappings into unpredictable locations. ASLR makes it more difficult for an attacker to redirect control to code that they have introduced into the address space of a process.

CC
Common Criteria for Information Technology Security Evaluation.

CEM
Common Evaluation Methodology for Information Technology Security Evaluation.

Credential
Data that establishes the identity of a user, e.g. a cryptographic key or password.

CSP
: Information that is either user or system defined and is used to operate a cryptographic module in processing encryption functions including cryptographic keys and authentication data, such as passwords, the disclosure or modification of which can compromise the security of a cryptographic module or the security of the information protected by the module.

DAR Protection
: Countermeasures that prevent attackers, even those with physical access, from extracting data from non-volatile storage. Common techniques include data encryption and wiping.

DEP
: An anti-exploitation feature of modern operating systems executing on modern computer hardware, which enforces a non-execute permission on pages of memory. DEP prevents pages of memory from containing both data and instructions, which makes it more difficult for an attacker to introduce and execute code.

Developer
: An entity that writes OS software. For the purposes of this document, vendors and developers are the same.

General Purpose Operating System
: A class of OSes designed to support a wide-variety of workloads consisting of many concurrent applications or services. Typical characteristics for OSes in this class include support for third-party applications, support for multiple users, and security separation between users and their respective resources. General Purpose Operating Systems also lack the real-time constraint that defines Real Time Operating Systems (RTOS). RTOSes typically power routers, switches, and embedded devices.

Host-based Firewall
: A software-based firewall implementation running on the OS for filtering inbound and outbound network traffic to and from processes running on the OS.

OS
: Software that manages physical and logical resources and provides services for applications. The terms *TOE* and *OS* are interchangeable in this document.

PII
: Any information about an individual maintained by an agency, including, but not limited to, education, financial transactions, medical history, and criminal or employment history and information which can be used to distinguish or trace an individual's identity, such as their name, social security number, date and place of birth, mother's maiden name, biometric records, etc., including any other personal information which is linked or linkable to an individual.

PP
: An implementation-independent set of security requirements for a category of products.

SAR
: A requirement to assure the security of the TOE.

Sensitive Data
: Sensitive data may include all user or enterprise data or may be specific application data such as PII, emails, messaging, documents, calendar items, and contacts. Sensitive data must minimally include credentials and keys. Sensitive data shall be identified in the OS's TSS by the ST author.

SFR
: A requirement for security enforcement by the TOE.

ST
A set of implementation-dependent security requirements for a specific product.

TOE
The product under evaluation. In this case, the Operating System and its supporting documentation.

TSF
The security functionality of the product under evaluation.

TSS
A description of how a TOE satisfies the SFRs in a ST.

User
A user is subject to configuration policies applied to the operating system by administrators. On some systems under certain configurations, a normal user can temporarily elevate privileges to that of an administrator. At that time, such a user should be considered an administrator.

8.3 References

CC — **Common Criteria for Information Technology Security Evaluation**
Version 3.1R5
Date April 2017
Location http://www.commoncriteriaportal.org/files/ccfiles/CCPART1V3.1R5.pdf
Location http://www.commoncriteriaportal.org/files/ccfiles/CCPART2V3.1R5.pdf
Location http://www.commoncriteriaportal.org/files/ccfiles/CCPART3V3.1R5.pdf

OSPP — **Protection Profile for General Purpose Operating Systems**
Version 4.2.1
Date 2019-04-22
Location https://www.niap-ccevs.org/Profile/Info.cfm?PPID=442&id=442

RFC4252 — **The Secure Shell (SSH) Authentication Protocol**
Date January 2006
Location http://tools.ietf.org/html/rfc4252

RFC4253 — **The Secure Shell (SSH) Transport Layer Protocol**
Date January 2006
Location http://tools.ietf.org/html/rfc4253

RFC5656 — **Elliptic Curve Algorithm Integration in the Secure Shell Transport Layer**
Date December 2009
Location http://tools.ietf.org/html/rfc5656

SSH — **Functional Package for Secure Shell (SSH)**
Version 1.0
Date 2021-05-13
Location https://www.niap-ccevs.org/Profile/Info.cfm?PPID=459&id=459