FIN.X RTOS SE V5

Security Target (Lite)

Revision History

Rev.	Change Order	Date	Changes Description
01	First Release	02.04.2021	--
02	16C10038157, 01	11/05/2022	Responses to ROAs

Common Criteria V3.1, rev. 5 – Security Target

Table of Contents

INDEX OF TABLES

INDEX OF FIGURES

1 Introduction

This is the Security Target Lite (ST-Lite) for the Computer Software Configuration Item (CSCI) identified as FIN.X RTOS SE V5, which is a linux-based operating system running on x86_64 machines.

This document is a "Lite" version of the FIN.X RTOS SE V5.0 Security Target, PN 16210068210, and it is developed in accordance to the rules stated in http://www.commoncriteriaportal.org/files/supdocs/CCDB-2006-04-004.pdf.

1.1 Security Target Identification

This ST is identified as:

- Title: **FIN.X RTOS SE V5.0 Security Target (Lite)**;
- Part number: **16210068210**;
- Revision: **02**;
- Date: **11/05/2022**.

1.2 TOE Identification

This ST applies to the CSCI identified as:

- Title: **FIN.X RTOS SE V5**;
- Abbreviation: **CSCI_FINXSE_V5,** or **TOE**.

The TOE's version is included in the TOE's identifier (title), i.e. version '5' (V5 or V5.0).

1.3 TOE Type

The TOE type is Linux-based general-purpose operating system.

1.4 TERMINOLOGY

External entities: include human users, as well as other IT systems.

Objects: objects belong to one of the following categories: file system objects, IPC objects, memory objects, and network objects. Processes are objects when they are the target of signal-related system calls.

- **FS objects** - File system objects:
  - ***ext4*** standard file system for general data;
  - ***XFS*** standard file system for general data;
  - ***VFAT*** special purpose file system for UEFI BIOS support mounted at /boot/efi;
  - ***iso9660*** ISO9660 file system for CD-ROM and DVD;
  - ***tmpfs*** the temporary file system backed by RAM;
  - ***rootfs*** the virtual root file system used temporarily during system boot;
  - ***procfs*** process file system holding information about processes, general statistical data and tunable kernel parameters;
  - ***sysfs*** system-related file system covering general information about resources maintained by the kernel including several tunable parameters for these resources;
  - ***devpts*** pseudoterminal file system for allocating virtual TTYs on demand;
  - ***devtmpfs*** temporary file system that allows the kernel to generate character or block device nodes;
  - ***binfmt_misc*** configuration interface allowing the assignment of executable file formats with user space applications;
  - ***securityfs*** interface for loadable security modules (LSM) to provide tunables and configuration interfaces to user space;
  - ***cgroup*** interface for configuring the control groups mechanism provided by the kernel;
  - ***debugfs*** interface for accessing low-level kernel data.

  Please note that the TOE supports a number of additional virtual (i.e. without backing of persistent storage) file systems, which are only accessible to the TSF - they are not or cannot be mounted. All above mentioned virtual file systems implement access decisions based DAC attributes inferred from the underlying process' DAC attributes.

- **IPC Objects** - Inter Process Communication (IPC) objects:
  - Semaphores
  - Shared memory
  - Message queues
  - Named pipes
  - UNIX domain socket special files
  - Signals.

- **Network objects** - irrespective of their type, such as Internet sockets and netlink sockets.
- **Storage device objects** - objects provided by block devices;
- **Named Objects** - objects that are subject to discretionary access control. This includes all objects except memory objects.
- **Public objects** - objects for which the TSF unconditionally permits all entities "read" access. Only the TSF or authorized administrators may create, delete, or modify the public objects.

Subjects: Processes acting on behalf of a human user or technical entity. There are two types of subjects:

- Untrusted internal subjects: they are processes running on behalf of some user, running outside the TSF;
- Trusted internal subjects - they are processes running as part of the TSF. Examples are service daemons and the process implementing the identification and authentication of users.

User: any person who interacts with the TOE. The authorized non-administrative user and authorized administrator are security roles maintained within the TOE. The TOE associate each user with roles. In this document, unless differently specified, "authorized user" include both administrative and non-administrative authorized users.

TSF data: one of the following objects

- TSF executable code;
- Subject meta data - All data used for subjects except data which is not interpreted by the TSF and does not implement parts of the TSF (this data is called user data);
- Named object meta data - All data used for the respective objects except data which is not interpreted by the TSF and does not implement parts of the TSF (this data is called user data);
- User accounts;
- Audit records;
- FIPS 186-4 [**FIPS186-4**] RSA, ECDSA, and DSA public/private key pairs and any associated passphrase;
- Session keys for accessing cryptographically-protected storage space and passphrases protecting those session keys.

User data: one of the following objects

- Non-TSF executable code used to drive the behavior of subjects;
- Data not interpreted by TSF and stored or transmitted using named objects;
- Keys for accessing cryptographically-protected storage space and any associated passphrase;
- FIPS 186-4 [**FIPS186-4**] RSA, ECDSA, and DSA public/private key pairs and any associated passphrase.

1.5 TOE OVERVIEW

The TOE is a real-time linux-based operating system derived from the Gentoo Linux distribution [**Gentoo**], whose strengths are its highly flexibility, scalability, configurability, and customizability, which make it very easy to optimize for embedded systems.

The TOE meets all functional requirements of the NIAP OSPP [**NIAP-OSPP**] along with the SSH and TLS functional packages [**SSH-FP**, **TLS-FP**].

1.5.1 Intended Method of Use

As multi-user and multi-tasking operating system, the TOE may provide services to several users at the same time. After successful login, the users have access to a general computing environment, allowing the start-up of user applications, issuing user commands at shell level, creating and accessing files. The TOE provides adequate mechanisms to separate the users and protect their data, as stated on section 1.6.3.7. Privileged commands are restricted to administrative users.

The TOE is intended to operate as a non-networked system or to operate in a networked environment with other instantiations of the TOE as well as other well-behaved peer systems operating in accordance with a a defined common security policy that does not conflict with this Security Target.

It is assumed that responsibility for the safeguarding of the data protected by the TOE can be delegated to the TOE human users, if such users are allowed to log on and spawn processes on their behalf. All data (within the defined logical boundaries) are under the control of the TOE. The data are stored in named objects, and the TOE can associate with each named object a description of the access rights to that object. The TOE enforces controls so that access to data objects can only take place in accordance with the access restrictions placed on that object by its owner, and by authorized administrators.

Human users, as well as services offered by the TOE, are assigned unique user identifiers. These user identifiers are used together with the attributes and roles assigned to the user identifier as the basis for access control decisions. The TOE authenticates the claimed identity of the user before allowing the user to perform any further actions. Authorized services may be spawned by the TOE without the need for user interaction. The TOE internally maintains a set of identifiers associated with processes, which are derived from the unique user identifier upon login of the user or from the configured user identifier for a TOE spawned service. Some of those identifiers may change during the execution of the process according to a policy implemented by the TOE.

Discretionary access rights (e.g. read, write, execute) can be assigned to data objects with respect to subjects identified with their UID and GID. Once a subject is granted access to an object, the content of that object may be freely used to influence other objects accessible to this subject.

1.5.2 Summary of the TOE security features

The primary security features of the TOE are specified as part of section 1.6.3.

These primary security features are supported by domain separation and reference mediation, which ensure that the features are always invoked and cannot be bypassed.

1.5.3 Required Hardware

The TOE does not require any specific hardware platform to operate. It works on all x86_64 Intel Architecture platforms, with minimum requirements of:

- UEFI Secure Boot
- Microprocessor: Intel Pentium i686 or Intel Xeon
- HD: SATA, 128GB
- RAM: 2GB
- Network adapter (only required for the TOE operating into a networked environment)
- CD/DVD drive, possibly accessible via USB interface (only required for the TOE installation)

The hardware used during the evaluation process is listed on Table 1-1:

Vendor	Model	CPU
Concurrent Technologies (https://www.gocct.com/)	Board VP B1x	Intel® Core™ i7
Panasonic (https://www.panasonic.com)	TOUGHBOOK CF-54	Intel® Core™ i7
Larimart (http://www.larimart.it/)	LRT-314	Intel® Core™ i7
Themis (https://www.mrcy.com)	Rugged Enterprise Server XR5	Intel® Xeon®

Table 1-1: TOE physical boundaries – Evaluated hardware

All tests conducted on these platforms have demonstrated that all security functions defined in this document work properly. The TOE provides a dedicated installation procedure to support the installation of the operating system on the above platforms.

1.6 TOE DESCRIPTION

1.6.1 Introduction

As general purpose, multi-user, and multi-tasking Linux based operating system, the TOE supports a broad range of system boards and computing systems, ranging from multi-processor workstations and servers to single board computers operating in ruggedized and embedded environments.

The TOE provides a platform for a variety of applications, and it provides a secure and predictable execution environment where interaction with external entities is only permitted via trusted channels. The TOE can be configured to run a kernel with the PREEMPT_RT [PREEMPT_RT] patch or a kernel without the PREEMPT_RT patch. When configured with real-time enhancers (i.e. by using the Linux Kernel PREEMPT_RT patch), the TOE will provide adequate support for applications with real-time requirements.

This product evaluation covers a potentially distributed network of systems running the evaluated versions and configurations of the TOE. The hardware platforms selected for the evaluation consist of machines which are available when the evaluation has completed and to remain available for a substantial period of time afterwards.

The TOE Security Functions (TSFs) consist of functions of the TOE that run in kernel mode plus some trusted processes. These are the functions that implement the security functional requirements defined in section 6.

The TOE includes standard networking applications, including applications allowing access of the TOE via cryptographically protected communication channels, such as SSH.

The hardware, the BIOS firmware and potentially other firmware layers between the hardware and the TOE are considered to be part of the TOE environment.

1.6.2 Physical Scope

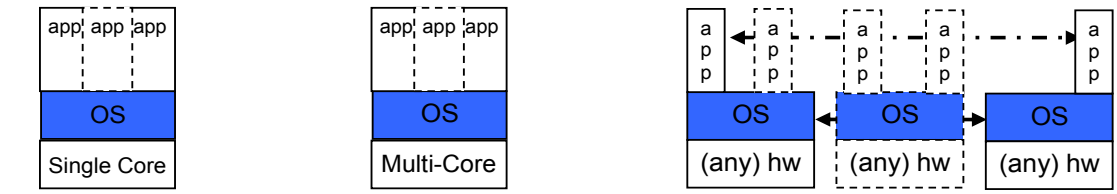
The TOE system software is the FIN.X RTOS SE V5.0 operating system as identified in section 1.2. The TOE and its documentation are supplied on ISO images and PDF files distributed via the TOE's owner Network or on a CD/DVD ROM.

The TOE includes a package holding the additional user and administrator documentation. In addition to the installation media, the following documentation is provided:

- Software User Manual for the CSCI FIN.X RTOS SE V5.0

The hardware that is applicable to the evaluated configuration is listed in section 1.5.3.

TOE Operational Environment: Figure 1 shows the possible deployment options that are available for the TOE considered in this evaluation. Configurations a) and b) cover the deployment of the TOE on a single and a multi-core platform. Configuration c) covers the distributed deployment of the TOE on a set of hardware platforms that may comprise any combinations of the configurations a) and b). All options support single or multiple software application deployment on top of the TOE.



a) Single core CPU, modular application deployment.

b) Multi-core CPU, modular application deployment.

c) Distributed deployment (hardware & s/w modularity variants all possible)

Figure 1 - Possible deployment options available for the TOE considered in this evaluation

When deployed according to the above configuration c), the TOE operate in a networked environment with other instantiations of the TOE as well as other well-behaved peer systems operating within the same management domain. All those systems are assumed to be configured in accordance with a defined common security policy that does not conflict with this Security Target.

1.6.3 Logical Scope

The primary security features of the TOE are described in the following paragraphs. Along with these security functions, the TOE provides many other functions and mechanisms. The evaluation ensures that all these additional functions do not interfere with the below mentioned security mechanisms in the evaluated configuration. Mechanisms and functions that would interfere with the operation of the security functions are disallowed in the evaluated configuration and the Secure Installation, Configuration & Operations Guide provides instructions to the authorized administrator on how to disable them.

1.6.3.1 Identification and Authentication

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

1. All individual users are assigned a unique user identifier within the single host system that forms the TOE. This user identifier is used together with the attributes and roles assigned to the user as the basis for access control decisions.
2. The TOE authenticates the claimed identity of the user before allowing the user to perform any further actions.
3. The TOE internally maintains a set of identifiers associated with processes, which are derived from the unique user identifier upon login of the user. Some of those identifiers may change during the execution of the process (e.g., using the *su* command) according to a policy implemented by the TOE.

The authentication security function allows the following authentication methods:

- Password-based authentication – It is provided by pluggable authentication modules (PAM) and it is always used during the log in process at the local console or at the remote console, and when becoming another user (i.e. during '*su*' actions). Password quality enforcement mechanisms are offered by the TOE. These mechanisms are enforced at the time when the password is changed;
- Public-key-based authentication – It is used for initiating a SSH access without supplying the user's password as specified by the SSH V2 protocol.

1.6.3.2 Audit

The TOE provides an audit capability that allows the generation of audit records for security critical events. The authorized administrator can select which events are audited and for which users auditing is active.

The TOE provides tools that help the authorized administrator to extract specific types of audit events, audit events for specific users, audit events related to specific file system objects, or audit events within a specific time frame from the overall audit records collected by the TOE. The audit records are stored in ASCII text, no conversion of the information into human readable form is necessary.

The audit system detects when the capacity of the audit trail exceeds configurable thresholds, and the authorized administrator can define actions to take when the threshold is exceeded.

1.6.3.3 Discretionary Access Control

Discretionary Access Control (DAC) restricts access to file system objects based on Access Control Lists (ACLs) that include the standard UNIX permissions for user, group and others. Access control mechanisms also protect IPC objects from unauthorized access.

The DAC mechanism is also used to ensure that untrusted users cannot tamper with the TOE mechanisms.

1.6.3.4 Object Reuse

File system objects as well as memory and IPC objects will be cleared before they can be reused by a process belonging to a different user.

1.6.3.5 Security Management

The security management facilities provided by the TOE are usable by authorized administrators to modify the configuration of TSF.

All authorized users are able to change their authentication data.

1.6.3.6 Cryptographic Services

The TOE provides cryptographically secured communication to allow remote entities to log into the TOE.

For interactive usage, an implementation of the SSH V2 protocol is provided.

The TOE implements Transport Layer Security v1.2 (or higher) to provide protected, authenticated, confidential, and tamper-proof networking between two peer computers.

The TOE provides strong cryptographic primitives [**FIPS140**, **FIPS186, OMSecPol**] that users and services can utilize for unspecified purposes.

The TOE conforms to LUKS standard [**LUKS, PBKDF2**] for supporting confidentiality of data storage via cryptographically-protected storage space: encrypted data can only be decrypted, e.g. accessed, by the user's session key.

1.6.3.7 TSF Protection

While in operation, the kernel software and data are protected by the hardware memory protection mechanisms. The memory and process management components of the kernel ensure a user process cannot access kernel storage or storage belonging to other processes.

Non-kernel TSF software and data are protected by DAC and process isolation mechanisms. In general, files and directories containing internal TSF data (e.g., configuration files) are also protected from reading by DAC permissions.

The TOE implements and enforces the following self-protection mechanisms that protect the security mechanisms of the TOE as well as software executed by the TOE:

- Address Space Layout Randomization for user space code;
- Stack buffer overflow protection using stack canaries;
- Secure Boot ensuring that the boot chain up to and including the kernel together with the boot image (initramfs) is not tampered with;

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

1.6.3.8 Additional Functions

The TOE provides other functions and mechanisms in addition to the evaluated ones such as:

- **Virtualization support**: The TOE offers full virtualization support allowing other operating systems to execute on the TOE. The Linux kernel operates as a hypervisor while the creation and run of guest operating systems is performed by QEMU, which executes as an unprivileged application from the view-point of the Linux kernel. Virtual machines are managed via the *libvritd* daemon, which run with privileges of the root user.
- **Linux Container support**: The TOE offers userspace virtualization support via Linux Container. To ensure Linux Containers do not interfere with the operation of the security functions of the TOE, support for unprivileged Linux Containers is provided. Linux Containers are managed via the *libvritd* daemon, which run with privileges of the root user.

Even if not part of the evaluation configuration, added functions and mechanisms are hardened for security.

1.6.3.9 Configurations

The evaluated configurations are defined as follows:

- The CC evaluated packages indicated in the Software Version Document and installed accordingly to the Software User Manual.

This mentioned package list includes packages that contribute to the TSF as well as packages that contain untrusted user programs that belongs to the FIN.X RTOS SE V5.0 distribution, but do not contribute or interfere with the TSF.

Deviations from the configurations and settings specified with the Software Version Document and/or the Software User Manual are not permitted.

1.6.3.10 TOE Environment

Several TOE systems may be interlinked in a network. If other systems are connected to the network, they need to be configured and managed by the same authority using an appropriate security policy that does not conflict with the security policy of the TOE. All links between this network and untrusted networks (e.g. the Internet) need to be protected by appropriate measures such as carefully configured firewall systems that prohibit attacks from the untrusted networks. Those protections are part of the TOE environment.

2 Conformance Claims

2.1 Conformance with CC parts 2 and 3

The TOE claims conformance to:

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

2.2 Conformance with Protection Profiles

The TOE claims exact conformance to:

- NIAP OSPP **[NIAP-OSPP]**, along with the following functional packages
  - Functional Package for SSH **[SSH-FP]**;
  - Functional Package for TLS **[TLS-FP]**.

2.2.1 Applied Technical Decisions

NIAP issued the some technical decisions for the OSPP and TLS, SSH functional packages. The following are the ones applied in this ST:

- 0588 – Session Resumption Support in TLS package
- 0578 – SHA-1 is no longer mandatory
- 0525 – Updates to Certificate Revocation (FIA_X509_EXT.1)
- 0501 – Cryptographic selections and updates for OS PP
- 0493 – X.509v3 certificates when using digital signatures for Boot Integrity
- 0463 – Clarification for FPT_TUD_EXT
- 0442 – Updated TLS Ciphersuites for TLS Package
- 0441 – Updated TLS Ciphersuites for OS PP
- 0365 – FCS_CKM_EXT.4 selections

2.2.2 Excluded Technical Decisions

The following NIAP technical decisions were excluded:

- 0600 – Conformance claim sections updated to allow for MOD_VPNC_V2.3. This TD was excluded as VPN functionality is not claimed for the TOE
- 0513 – CA Certificate loading. This TD was excluded as TOE is not able to manage trust store
- 0499 – Testing with pinned certificates. This TD was excluded as pinned certificates are not supported by the TOE
- 0496 – GPOS PP adds allow-with statement for VPN Client V2.1. This TD was excluded as VPN functionality is not claimed for the TOE
- 0469 – Modification of test activity for FCS_TLSS_EXT.1.1 test 4.1. This TD was excluded as change does not affect assurance activity because the TOE supports the latest TLS version and TD issue does not apply
- 0386 – Platform-Provided Verification of Update. This TD was excluded as verification of updates is not performed by the TOE via platform-provided functions

3 Security Problem Definition

This section defines the expected TOE security environment in terms of the threats, security assumptions, and the security policies that must be followed for the TOE and the related IT environment.

3.1 THREATS

Threats to be countered by the TOE are characterized by the combination of an asset being subject to a threat, a threat agent and an adverse action.

3.1.1 Assets

Assets to be protected are:

- Persistent storage objects used to store user data and/or TSF data, where this data needs to be protected from any of the following operations:
  - Unauthorized read access
  - Unauthorized modification
  - Unauthorized deletion of the object
  - Unauthorized creation of new objects
  - Unauthorized management of object attributes
- Transient storage objects, including network data.
- TSF functions and associated TSF data.
- The resources managed by the TSF that are used to store the above-mentioned objects, including the metadata needed to manage these objects.

3.1.2 Threat Agents

Threat agents are external entities that potentially may attack the TOE. They satisfy one or more of the following criteria:

- External entities not authorized to access assets may attempt to access them either by masquerading as an authorized entity or by attempting to use TSF services without proper authorization.
- External entities authorized to access certain assets may attempt to access other assets they are not authorized to either by misusing services they are allowed to use or by masquerading as a different external entity.
- Untrusted subjects may attempt to access assets they are not authorized to either by misusing services they are allowed to use or by masquerading as a different subject.

Threat agents are typically characterized by a number of factors, such as expertise, available resources, and motivation, with motivation being linked directly to the value of the assets at stake.

The TOE protects against intentional and unintentional breach of TOE security by attackers possessing a basic attack potential.

3.1.3 Threats countered by the TOE

The following are known or presumed threats to assets that are addressed by the TOE based on conformance to the NIAP OSPP **[NIAP-OSPP]**:

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

There are not other known or presumed threats to assets that are addressed by the TOE based on other Functional Packages [**SSH-FP**, **TLS-FP**].

3.2 Organizational Security Policy

There are no Organizational Security Policies for the protection profile or functional package.

3.3 Security Assumptions

The specific conditions below are assumed to exist in a TOE environment conformant to the NIAP OSPP **[NIAP-OSPP]**:

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

There are not other conditions assumed to exist in the TOE environment.

4 Security Objectives

This section defines the security objectives for the TOE and its supporting environment. Security objectives, categorized as either TOE security objectives or objectives by the supporting environment, reflect the stated intent to counter identified threats, comply with any organizational security policies identified, or address identified assumptions. All of the identified threats, organizational policies (if any), and assumptions are addressed under one of the categories below.

4.1 Security Objectives for the TOE

Security objectives for the TOE are next identified with "O." appended to the beginning of the name and the objective. The following security objectives for the TOE are needed to comply with the NIAP OSPP **[NIAP-OSPP]**:

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

SSH and TLS Functional Packages [**SSH-FP**, **TLS-FP**] do not define any threat nor security objective to supplement the ones in the NIAP OSPP [**NIAP-OSPP**].

4.2 SECURITY OBJECTIVES FOR THE OPERATIONAL ENVIRONMENT

The TOE is assumed to be complete and self-contained and, as such, is not dependent upon any other products to perform properly. However, certain objectives with respect to the general operating environment must be met.

Security objectives for the operational environment of the TOE are next identified with "OE." appended to the beginning of the name and the objective.

The following security objectives for the operational environment of the TOE are specified by the NIAP OSPP **[NIAP-OSPP]**:

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

There are not other security objectives for the operational environment of the TOE.

4.3 SECURITY OBJECTIVES COVERAGE AND RATIONALE

Table 4-1 provides a mapping of TOE threats to objectives, showing that each threat is countered by a security objective at least. A rationale provides justification that the security objectives are suitable to counter each assigned threat.

Threats	Security Objectives	Consistency Rationale
T.NETWORK_ATTACK	O.PROTECTED_COMMS, O.INTEGRITY, O.MANAGEMENT, O.ACCOUNTABILITY	The threat T.NETWORK_ATTACK is countered by O.PROTECTED_COMMS as this provides for integrity of transmitted data. 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	O.PROTECTED_COMMS, O.MANAGEMENT	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	O.INTEGRITY, O.ACCOUNTABILITY	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	O.PROTECTED_STORAGE	The objective O.PROTECTED_STORAGE protects against unauthorized attempts to access physical storage used by the TOE.

Table 4-1: Mapping of threats to security objectives

Table 4-2 provides a mapping of assumptions to the Operational Environment objectives, showing that each assumption is covered by a operational environment objective at least. A rationale provides justification that the security objectives for the environment are suitable to cover each individual assumption.

Assumptions	Operational Environment objectives	Consistency Rationale
A.PLATFORM	OE.PLATFORM	The operational environment objective OE.PLATFORM is realized through A.PLATFORM.
A.PROPER_USER	OE.PROPER_USER	The operational environment objective OE.PROPER_USER is realized through A.PROPER_USER.

Assumptions	Operational Environment objectives	Consistency Rationale
A.PROPER_ADMIN	OE.PROPER_ADMIN	The operational environment objective OE.PROPER_ADMIN is realized through A.PROPER_ADMIN.

Table 4-2: Mapping of assumptions to security objectives for the Operational Environment

5 Extended Functional Requirements

The definition of all SFRs with the appendix of "_EXT" is supplied by the NIAP OSPP [**NIAP-OSPP**] along with Functional Packages [**SSH-FP**, **TLS-FP**]. These SFRs are not defined in this section.

6 Security Functional Requirements

This section contains detailed security functional requirements for the TOE.

All SFRs are derived from the the NIAP OSPP [**NIAP-OSPP**] along with Functional Packages [**SSH-FP**, **TLS-FP**].

The following notations are used:

- Assignment or selection: Indicated with *italicized* text;
- Refinement: Indicated with **bold** text for additions to the text, and with ~~strikethrough text~~ for deletions from the text;
- Iteration: Indicated by appending a suffix to the original SFRsuggesting the purpose of the iteration, e.g. '(SSH) for an SFR relating to SSH functionality, and/or a sequential number in parentheses, e.g. (1).

6.1 Security Functional Requirements

6.1.1 Cryptographic Support

6.1.1.1 Cryptographic key generation (Refined)

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

- ***RSA schemes using cryptographic key sizes of 2048-bit or greater that meet the following: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.3,***
- ***ECC schemes using "NIST curves" P-256, P-384 and P-521, that meet the following: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.4,***
- ***FFC schemes using cryptographic key sizes of 2048-bit or greater that meet the following: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.1,***
- ***FFC Schemes using safe primes that meet the following: NIST Special Publication 800-56A Revision 3, "Recommendation for Pair-Wise Key Establishment Schemes",***
- ***FFC Schemes using Diffie-Hellman group 14 that meet the following: RFC 3526.***

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

Application Note: *FCS_CKM.1.1 applies the NIAP-OSPP technical decision "0501 – Cryptographic selections and updates for OS PP".*

Application Note: *The TOE supports the generation of RSA, DSA and ECDSA keys for the OpenSSH host key as well as the OpenSSH user keys using the ssh-keygen(1) application*.

Application Note: *The TOE supports the generation of RSA, DSA and ECDSA keys for the host authentication used as part of the TLS protocol using the openssl(1)*

6.1.1.2 Cryptographic Key Establishment (Refined)

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

- ***RSA-based key establishment schemes that meets the following: RSAES-PKCS1-v1_5 as specified in Section 7.2 of RFC 8017, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.2,***
- ***Elliptic curve-based key establishment schemes that meets the following: NIST Special Publication 800-56A Revision 3, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography",***
- ***Finite field-based key establishment schemes that meets the following: NIST Special Publication 800-56A Revision 3, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography",***
- ***Key establishment scheme using Diffie-Hellman group 14 that meets the following: RFC 3526.***

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

Application Note: *FCS_CKM.2.1 applies the NIAP-OSPP technical decision "0501 – Cryptographic selections and updates for OS PP".*

Application Note*: The TOE performs key agreement for the SSH protocol using the OpenSSL library.*

Application Note*: As part of the key agreement for SSHv2, the OpenSSH client and server applications implement the key derivation function (KDF) according to SP800-135*

6.1.1.3 Cryptographic Key Destruction

FCS_CKM_EXT.4.1 The OS shall destroy cryptographic keys in accordance with the specified cryptographic key destruction methods:

- *For volatile memory, the destruction shall be executed by a*
  - *single overwrite consisting of zeroes,*
- *For non-volatile memory that consists of the invocation of an interface provided by the underlying platform that*
  - *Logically addresses the storage location of the key and performs at least one overwrite consisting of zeros.*

Application Note: *FCS_CKM_EXT.4.1 applies the NIAP-OSPP technical decision "0365 – FCS_CKM_EXT.4 selections".*

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

6.1.1.4 Cryptographic Operation - Encryption/Decryption (Refined)

FCS_COP.1.1(SYM) The **OS** shall perform *encryption/decryption services for data* in accordance with a specified cryptographic algorithm:

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

  And,
- ***AES-GCM (as defined in NIST SP 800-38D),***
- ***No other modes***

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

Application *Note*: *FCS_COP.1(SYM) corresponds to FCS_COP.1(1) in the NIST OSPP [**NIST-OSPP**].*

Application Note: *The TOE performs symmetric encryption and decryption for the SSH and TLS protocols using the OpenSSL library.*

Application Note: *The TOE performs symmetric encryption and decryption part of the block device encryption using the Linux kernel crypto API.*

6.1.1.5 Cryptographic Operation - Hashing (Refined)

Application Note: *FCS_COP.1(HASH) corresponds to FCS_COP.1(2) in the NIST OSPP [**NIST-OSPP**].*

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

- *SHA-1,*
- *SHA-256,*
- *SHA-384,*
- *SHA-512*

And message digest sizes:

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

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

Application Note: *FCS_COP.1.1(HASH) applies the NIAP-OSPP technical decision "0578 – SHA-1 is no longer mandatory".*

Application Note: *The TOE performs hashing for the SSH protocol using the OpenSSL library.*

Application Note: *The TOE performs hashing to support the ESSIV initialization vector derivation from a sector number for CBC-based disk encryption.*

6.1.1.6 Cryptographic Operation - Signing (Refined)

Application Note: *FCS_COP.1(SIGN) corresponds to FCS_COP.1(3) in the NIST OSPP [**NIST-OSPP**].*

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

- ***RSA schemes using cryptographic key sizes of 2048-bit or greater that meet the following: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Section 4,***
- ***ECDSA schemes using "NIST curves" P-256, P-384 and P-521 that meet the following: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Section 5.***

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

Application Note: *The TOE performs signature operation for the SSHv2 protocols as well as the trusted update signature verification using the OpenSSL library.*

6.1.1.7 Cryptographic Operation - Keyed-Hash Message Authentication (Refined)

***Application Note**: FCS_COP.1(HMAC) corresponds to FCS_COP.1(4) in the NIST OSPP [**NIST-OSPP**].*

FCS_COP.1.1(HMAC) The **OS** shall perform *keyed-hash message authentication services* in accordance with a specified cryptographic algorithm:

- ***SHA-1,***
- ***SHA-256,***
- ***SHA-384,***
- ***SHA-512.***

with key sizes *equal to the message digest size of the chosen cipher* and message digest sizes

- *160 bits,*
- *256 bits,*
- *384 bits,*
- *512 bits.*

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

Application Note: *The TOE generates MACs for the SSH protocol using the OpenSSL library.*

6.1.1.8 Random Bit Generation

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

- *CTR_DRBG (AES)*

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

- *platform-based noise source*

with a minimum of

- *256 bits*

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

***Application Note**: The TOE generates random bits for the SSH protocol using the OpenSSL library.*

6.1.1.9 Storage of Sensitive Data

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

6.1.1.10 TLS Client Protocol

FCS_TLSC_EXT.1.1 The OS shall implement TLS 1.2 (RFC 5246) supporting the following cipher suites:

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

***Application Note:** FCS_TLSC_EXT.1.1 applies the NIAP-OSPP technical decision "0441 – Updated TLS Ciphersuites for OS PP".*

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

FCS_TLSC_EXT.1.3 The OS shall only establish a trusted channel if the peer certificate is valid.

6.1.1.11 TLS Client Curves Allowed

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

6.1.2 User Data Protection

6.1.2.1 Access Controls for Protecting User Data

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

6.1.3 Security Management

6.1.3.1 Management of security functions behaviour

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.

6.1.3.2 Specification of Management Functions

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

Management Function	Administrator	User
Enable/disable *screen lock*	X	o
Enable/disable *screen lock timeout*	X	o
Configure screen lock inactivity timeout	X	o
Configure local audit storage capacity	X	o
Configure minimum password Length	X	o
Configure minimum number of special characters in password	X	o
Configure minimum number of numeric characters in password	X	o
Configure minimum number of uppercase characters in password	X	o
Configure minimum number of lowercase characters in password	X	o
Configure lockout policy for unsuccessful authentication attempts through: • *timeouts between attempts,* • *Limiting number of attempts during a time period.*	X	o
Configure host-based firewall	X	o
Configure name/address of directory server with which to bind	X	o
Configure name/address of remote management server from which to receive management settings	X	o
Configure name/address of audit/logging server to which to send audit/logging records	X	o
Configure audit rules	X	o
Configure name/address of network time server	X	o
Enable/disable automatic software update	o	o
Configure WiFi interface	o	o

Management Function	Administrator	User
Enable/disable Bluetooth interface	o	o
Enable/disable *network interface cards*	X	o
Enable/disable *USB interfaces*	X	o
Enable/disable *no other external interfaces*	o	o
Configure and manage user accounts	X	o
No other functions	o	o

Table 6-1: Management Functions

6.1.4 Protection of the TSF

6.1.4.1 Access Controls

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

- Kernel and its drivers/modules,
- Security audit logs,
- Shared libraries,
- System executables,
- System configuration files,
- *no other object*

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

- Security audit logs,
- System-wide credential repositories,
- *no other object*

6.1.4.2 Address Space Layout Randomization

FPT_ASLR_EXT.1.1 The OS shall always randomize process address space memory locations with *at least 8* bits of entropy except for *non-Position-Independent-Executable applications and non-Position-Intependent-Code shared libraries*.

6.1.4.3 Stack Buffer Overflow Protection

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

6.1.4.4 Boot Integrity

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

- *no other executable code*

prior to its execution through the use of

- *a digital signature using an X509 certificate with hardware-based protection.*

Application Note: *FPT_TST_EXT.1.1 applies the NIAP-OSPP technical decision "0493 – X.509v3 certificates when using digital signatures for Boot Integrity".*

6.1.4.5 Trusted Update

Application Note*: FCS_COP.1(SIGN) corresponds to FCS_COP.1(3) in the NIST OSPP [**NIST-OSPP**].*

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.1.1(SIGN) to validate the authenticity of the response.

Application Note: *FPT_TUD_EXT.1.1 applies the NIAP-OSPP technical decision "0463 – Clarification for FPT_TUD_EXT".*

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.1.1(SIGN).

6.1.4.6 Trusted Update for Application Software

FPT_TUD_EXT.2.1 The OS shall provide the ability to check for updates to application software and shall use a digital signature scheme specified in FCS_COP.1.1(SIGN) to validate the authenticity of the response.

Application Note: *FPT_TUD_EXT.2.1 applies the NIAP-OSPP technical decision "0463 – Clarification for FPT_TUD_EXT".*

FPT_TUD_EXT.2.2 The OS shall cryptographically verify the integrity of updates to applications using a digital signature specified by FCS_COP.1.1(SIGN) prior to installation.

6.1.5 Audit Data Generation

6.1.5.1 Audit Data Generation (Refined)

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

a) Start-up and shutdown of the audit functions,

b) All auditable events for the *not specified* level of audit

and

c) Authentication events (Success/Failure),

d) Use of privileged/special rights events (Successful and unsuccessful security, audit, and configuration changes),

e) Privilege or role escalation events (Success/Failure),

f) File and object events (Successful and unsuccessful attempts to create, access, delete, modify, modify permissions),

g) User and Group management events (Successful and unsuccessful add, delete, modify, disable, enable, and credential change),

h) Audit and log data access events (Success/Failure),

i) Attempted application invocation with arguments (Success/Failure e.g. due to software restriction policy),

j) Kernel module loading and unloading events (Success/Failure),

k) Administrator or root-level access events (Success/Failure),

l) Establishment of SSH connection,

m) Termination of SSH connection session,

n) no other events.

Application Note: *The following auditable events applied from the SSH Functional Package [SSH-FP]:*

Requirement	Auditable Event	Additional Audit Record Contents
FCS_SSH_EXT.1	*Establishment of SSH connection*	*None*
FCS_SSH_EXT.1	*Termination of SSH connection session*	*None*
FCS_SSH_EXT.1	*None*	*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, *no other information*.

6.1.6 Identification and Authentication

6.1.6.1 Authentication failure handling (Refined)

FIA_AFL.1.1 The **OS** shall detect *an administrator configurable positive integer within [0 (disabled) – 65,535]* unsuccessful authentication attempts occur related to **events with**:

- ***Authentication based on user name and password***.

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

- ***Temporary (via administrator configurable seconds) Account Lockout for administrators,***
- ***Permanent Account Lockout for other users.***

6.1.6.2 Multiple Authentication Mechanisms (Refined)

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

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

to support user authentication.

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

- *Authentication on the local console is based on user name and password,*
- *Authentication via the SSHv2 protocol first performs the public-key-based authentication which is followed by the user name and password authentication if the public-key-based authentication was unsuccessful*

6.1.6.3 X.509 Certificate Validation

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

- RFC 5280 certificate validation and certificate path validation;
- The certificate path must terminate with a trusted CA certificate;
- The OS shall validate a certificate path by ensuring the presence of the basicConstraints extension, that the CA flag is set to TRUE for all CA certificates, and that any path constraints are met;
- The TSF shall validate that any CA certificate includes caSigning purpose in the key usage field;
- The OS shall validate the revocation status of the certificate using *OCSP as specified in RFC 6960, CRL as specified in RFC 5759***;**
- The OS shall validate the extendedKeyUsage field according to the following rules:
    - Certificates used for trusted updates and executable code integrity verification shall have the Code Signing Purpose (id-kp 3 with OID 1.3.6.1.5.5.7.3.3) in the extendedKeyUsage field;
    - Server certificates presented for TLS shall have the Server Authentication purpose (id-kp 1 with OID 1.3.6.1.5.5.7.3.1) in the extendedKeyUsage field;
    - Client certificates presented for TLS shall have the Client Authentication purpose (id-kp 2 with OID 1.3.6.1.5.5.7.3.2) in the EKU field;
    - S/MIME certificates presented for email encryption and signature shall have the Email Protection purpose (id-kp 4 with OID 1.3.6.1.5.5.7.3.4) in the EKU field;
    - OCSP certificates presented for OCSP responses shall have the OCSP Signing Purpose (id-kp 9 with OID 1.3.6.1.5.5.7.3.9) in the EKU field;
    - Server certificates presented for EST shall have the CMC Registration Authority (RA) purpose (id-kp-cmcRA with OID 1.3.6.1.5.5.7.3.28) in the EKU field. (Conditional).

Application Note: *FIA_X509_EXT.1.1 applies the NIAP-OSPP technical decision "525 – Updates to Certificate Revocation (FIA_X509_EXT.1)".*

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.

6.1.6.4 X.509 Certificate Authentication

FIA_X509_EXT.2.1 The OS shall use X.509v3 certificates as defined by RFC 5280 to support authentication for TLS and *no other protocols* connections.

6.1.6.5 FTA_TAB.1 Default TOE access banners

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

6.1.7 Trusted Path/Channel

6.1.7.1 Trusted channel communication

FTP_ITC_EXT.1.1 The OS shall use:

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

to provide a trusted communication channel between itself and authorized IT entities supporting the following capabilities:

- *audit server, authentication server, management server* 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.

6.1.7.2 Trusted Path

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

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

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

6.1.7.3 Secure Shell

6.1.7.3.1 SSH Protocol

FCS_SSH_EXT.1.1 The TOE shall implement SSH acting as a *client*, *server* in accordance with that complies with RFCs 4251, 4252, 4253, 4254, 4256, *5656, 6668, 8268, 8308, 8332* and *no other standard*.

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

- *"password" (RFC 4252),*
- *"keyboard-interactive" (RFC 4256),*

- *"publickey" (RFC 4252):*
  - *ssh-rsa (RFC 4253),*
  - *rsa-sha2-256 (RFC 8332),*
  - *rsa-sha2-512 (RFC 8332),*
  - *ecdsa-sha2-nistp256 (RFC 5656),*
  - *ecdsa-sha2-nistp384 (RFC 5656),*
  - *ecdsa-sha2-nistp521 (RFC 5656)*

and no other methods.

FCS_SSH_EXT.1.3 The SSH client shall ensure that, as described in RFC 4253, packets greater than *261,888 bytes* in an SSH transport connection are dropped.

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

- *aes128-ctr (RFC 4344),*
- *aes256-ctr (RFC 4344),*
- *aes128-cbc (RFC 4253),*
- *aes256-cbc (RFC 4253)*

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)*

and no other mechanisms.

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

- *diffie-hellman-group14-sha256 (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.

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:*
  - *ssh-rsa (RFC 4253),*
  - *rsa-sha2-256 (RFC 8332),*
  - *rsa-sha2-512 (RFC 8332),*
  - *ecdsa-sha2-nistp256 (RFC 5656),*
  - *ecdsa-sha2-nistp384 (RFC 5656),*
  - *ecdsa-sha2-nistp521 (RFC 5656)*

as described in RFC 4251 section 4.1.

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

- *ssh-rsa (RFC 4253),*
- *rsa-sha2-256 (RFC 8332),*
- *rsa-sha2-512 (RFC 8332),*
- *ecdsa-sha2-nistp256 (RFC 5656),*
- *ecdsa-sha2-nistp384 (RFC 5656),*
- *ecdsa-sha2-nistp521 (RFC 5656)*

6.1.7.4 Transport Layer Security

6.1.7.4.1 TLS Protocol

***Application Note**: FCS_TLS_EXT.1.1 corresponds to FCS_TLS_EXT.1.1 in the TLS extended package [**TLS-FP**].*

FCS_TLS_EXT.1.1 The product shall implement:

- *TLS as a client,*
- *TLS as a server*

***Application Note**: Applications may implement the TLS protocol – server side or client side – using the openssl(1) libraries.*

6.1.7.4.2 TLS Client Protocol

***Application Note**: FCS_TLSC_EXT.1(TLS) corresponds to FCS_TLSC_EXT.1 in the TLS functional package [**TLS-FP**].*

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

- *TLS_RSA_WITH_AES_128_CBC_SHA as defined in RFC 5246,*
- *TLS_RSA_WITH_AES_256_CBC_SHA as defined in RFC 5246,*

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

and also supports functionality for:

- *Mutual authentication.*

Application Note: *FCS_TLSC_EXT.1.1(TLS) applies the NIAP-TLS technical decision "0442: Updated TLS Ciphersuites for TLS Package".*

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

FCS_TLSC_EXT.1.3(TLS) The product shall not establish a trusted channel if the server certificate is invalid *with no exceptions*.

6.1.7.4.2.1 TLS Client Support for Mutual Authentication

Application Note*: FCS_TLSC_EXT.2.1(TLS) corresponds to FCS_TLSC_EXT.2.1 in the TLS functional package [**TLS-FP**].*

FCS_TLSC_EXT.2.1(TLS) The product shall support mutual authentication using X.509v3 certificates.

6.1.7.4.2.2 TLS Client Support for Signature Algorithms Extension

Application Note*: FCS_TLSC_EXT.3 corresponds to FCS_TLSC_EXT.3 in the NIAP OSPP [**NIAP-OSPP**].*

FCS_TLSC_EXT.3.1 The product shall present the signature_algorithms extension in the Client Hello with the supported_signature_algorithms value containing the following hash algorithms:

- *SHA256,*
- *SHA384,*
- *SHA512*

and no other hash algorithms.

6.1.7.4.2.3 TLS Client Protocol

***Application Note**: FCS_TLSC_EXT.4 corresponds to FCS_TLSC_EXT.4 in the NIAP OSPP [**NIAP-OSPP**].*

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

6.1.7.4.2.4 TLS Client Support for Supported Groups Extension

***Application Note**: FCS_TLSC_EXT.5 corresponds to FCS_TLSC_EXT.5 in the TLS functional package [**TLS-FP**].*

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

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

6.1.7.4.3 TLS Server Protocol

***Application Note**: FCS_TLSS_EXT.1 corresponds to FCS_TLSS_EXT.1 in the TLS functional package [**TLS-FP**].*

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

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

and also supports functionality for:

- *Mutual authentication.*

***Application Note:** FCS_TLSS_EXT.1.1 applies the NIAP-SSH technical decision "0442: Updated TLS Ciphersuites for TLS Package".*

Application Note: *FCS_TLSS_EXT.1.1 applies the NIAP-SSH technical decision "0588: Session Resumption Support in TLS package".*

FCS_TLSS_EXT.1.2 The product shall deny connections from clients requesting SSL 2.0, SSL 3.0, TLS 1.0 and *TLS 1.1*.

FCS_TLSS_EXT.1.3 The product shall perform key establishment for TLS using:

- *RSA with size:*
  - *2048 bits,*
  - *3072 bits,*
  - *4096 bits,*
  - *no other sizes*
- *Diffie-Hellman parameters with size:*
  - *2048 bits,*
  - *3072 bits,*
  - *4096 bits,*
  - *6144 bits, 8192 bits,*
  - *no other sizes*
- *ECDHE parameters using elliptic curves:*
  - *secp256r1,*
  - *secp384r1,*
  - *secp521r1,*
  - *and no other curves*
- *no other key establishment methods*

6.1.7.4.3.1 TLS Server Support for Mutual Authentication

Application Note*: FCS_TLSS_EXT.2 corresponds to FCS_TLSS_EXT.2 in the TLS functional package [**TLS-FP**].*

FCS_TLSS_EXT.2.1 The product shall support mutual authentication using X.509v3 certificates.

FCS_TLSS_EXT.2.2 The product shall not establish a trusted channel if the client certificate is invalid.

FCS_TLSS_EXT.2.3 The product shall not establish a trusted channel if the Distinguished Name (DN) or Subject Alternative Name (SAN) contained in a certificate does not match one of the expected identifiers for the client.

6.1.7.4.3.2 TLS Server Support for Signature Algorithms Extension

Application Note*: FCS_TLSS_EXT.3 corresponds to FCS_TLSS_EXT.3 in the TLS functional package [**TLS-FP**].*

FCS_TLSS_EXT.3.1 The product shall present the HashAlgorithm enumeration in supported_signature_algorithms in the Certificate Request with the following hash algorithms:

- *SHA256,*
- *SHA384,*

- *SHA512*

and no other hash algorithms.

6.2 RATIONALE FOR SECURITY FUNCTIONAL REQUIREMENTS

Table 6-2 provides a mapping of NIAP OSPP [**NIAP-OSPP**] security objectives to SFRs, showing that a security objective is addressed by a security functional requirement at least.

Security Objectives	Security Functional Requirements
O.ACCOUNTABILITY	***Addressed by***: FAU_GEN.1, FTP_ITC_EXT.1 ***Rationale***: FAU_GEN.1 defines the auditable events that must be generated to diagnose the cause of unexpected system behavior. FTP_ITC_EXT.1 provides a mechanism for the TSF to transmit the audit data to a remote system.
O.INTEGRITY	***Addressed by***: FPT_SBOP_EXT.1, FPT_ASLR_EXT.1, FPT_TUD_EXT.1, FPT_TUD_EXT.2, FCS_COP.1.1(HASH), FCS_COP.1.1(SIGN), FCS_COP.1.1(HMAC), FPT_ACF_EXT.1, FIA_X509_EXT.1, FPT_TST_EXT.1, FTP_ITC_EXT.1, FIA_AFL.1, FIA_UAU.5 ***Rationale***: FPT_SBOP_EXT.1 enforces stack buffer overflow protection that makes it more difficult to exploit running code. FPT_ASLR_EXT.1 prevents attackers from exploiting code that executes in static known memory locations. FPT_TUD_EXT.1 and FPT_TUD_EXT.2 enforce integrity of software updates. FCS_COP.1.1(HASH), FCS_COP.1.1(SIGN), and FCS_COP.1.1(HMAC) provide the cryptographic mechanisms that are used to verify integrity values. FPT_ACF_EXT.1 guarantees the integrity of critical components by preventing unauthorized modifications of them. FIA_X509_EXT.1 provides X.509 certificates as a way of validating software integrity. FPT_TST_EXT.1 verifies the integrity of stored code. FIA_UAU.5 provides mechanisms that prevent untrusted users from accessing the TSF and FIA_AFL.1 prevents brute-force authentication attempts. FTP_ITC_EXT.1 provides trusted remote communications which makes a remote authenticated session less susceptible to compromise.
O.MANAGEMENT	***Addressed by***: FMT_MOF_EXT.1, FMT_SMF_EXT.1, FTA_TAB.1, FTP_TRP.1 ***Rationale***: FMT_SMF_EXT.1 defines the TOE's management functions and FMT_MOF_EXT.1 defines the privileges required to invoke them. FTP_TRP.1 provides one or more secure remote interfaces for management of the TSF and FTA_TAB.1 provides actionable warnings against misuse of these interfaces.
O.PROTECTED_STORAGE	***Addressed by***: FCS_STO_EXT.1, FCS_RBG_EXT.1, FCS_COP.1.1(SYM), FDP_ACF_EXT.1 , FCS_CKM_EXT.4 ***Rationale***: FCS_STO_EXT.1 provides a mechanism by which the TOE can designate data as 'sensitive' and subsequently require it to be encrypted. FCS_COP.1.1(SYM) defines the symmetric algorithm used to encrypt and decrypt sensitive data. FCS_RBG_EXT.1 defines the random bit generator used to create the symmetric keys used to perform this encryption and decryption. FDP_ACF_EXT.1 enforces logical access control on stored data. FCS_CKM_EXT.4 ensure there cannot be unauthorized access to encrypted data by users that successfully recoved any removed (but still valid) LUKS authentication key file.
O.PROTECTED_COMMS	***Addressed by***: FCS_TLSC_EXT.1, FCS_TLSC_EXT.2, FCS_TLSC_EXT.3, FCS_TLSC_EXT.4, FCS_RBG_EXT.1, FCS_CKM.1, FCS_CKM.2, FCS_CKM_EXT.4, FCS_COP.1.1(SYM), FCS_COP.1.1(HASH), FCS_COP.1.1(SIGN), FCS_COP.1.1(HMAC), FIA_X509_EXT.1, FIA_X509_EXT.2, FTP_ITC_EXT.1

Security Objectives	Security Functional Requirements
	Rationale: FCS_TLSC_EXT.1 FCS_TLSC_EXT.2, FCS_TLSC_EXT.3, and FCS_TLSC_EXT.4 define the ability of the TOE to act as a TLS client as a method of enforcing protected communications. FCS_CKM.1, FCS_CKM.2, FCS_COP.1.1(SYM), FCS_COP.1.1(HASH), FCS_COP.1.1(SIGN), FCS_COP.1.1(HMAC), and FCS_RBG_EXT.1 define the cryptographic operations and key lifecycle activity used to support the establishment of protected communications. FIA_X509_EXT.1 defines how the TSF validates x.509 certificates as part of establishing protected communications. FIA_X509_EXT.2 defines the trusted communication protocols for which the TOE must perform certificate validation operations. FTP_ITC_EXT.1 defines the trusted communications channels supported by the TOE. FCS_CKM_EXT.4 ensure there cannot be unauthorized connections by masquerade users that successfully recoved any removed (but still valid) SSH authentication key. It also ensure there cannot be unauthorized connections by remote TLS client that successfully recoved any removed (but still valid) TLS authentication key.

Table 6-2: Mapping of NIAP OSPP [NIAP-OSPP] security objectives to SFRs

Security Objectives	Security Functional Requirements
O.PROTECTED_COMMS	***Addressed by***: FCS_SSH_EXT.1 ***Rationale***: FCS_SSH_EXT.1 define the ability of the TOE to use the SSH protocol as a method of enforcing protected communications.

Table 6-3: Rational for the NIAP SSH EP [SSH-FP] SFRs

Security Objectives	Security Functional Requirements
O.PROTECTED_COMMS	***Addressed by***: FCS_TLS_EXT.1, FCS_TLSC_EXT.1, FCS_TLSC_EXT.2, FCS_TLSC_EXT.5, FCS_TLSS_EXT.1, FCS_TLSS_EXT.2, FCS_TLSS_EXT.3 ***Rationale***: FCS_TLS_EXT.1, FCS_TLSC_EXT.1, and FCS_TLSS_EXT.1 define the ability of the TOE to use the TLS protocol as a method of enforcing protected communications. FCS_TLSC_EXT.2 and FCS_TLSS_EXT.2 provide support for TLS client and server mutual authentication. FCS_TLSS_EXT.3 provides support for TLS client and server signature algorithms extension. FCS_TLSC_EXT.5 provides support for supported groups extension i.e. the elliptic curve supported groups.

Table 6-4: Rational for the NIAP TLS EP [TLS-FP] SFRs

7 Security Assurance Requirements

Security Assurance Requirements are specified in the Protection Profile [**NIAP-OSPP**] and the functional packages [**SSH-FP**, **TLS-FP**]. Then, they are not re-iterated in this document.

7.1 Security Assurance Requirements Rationale

This Security Target claims exact compliance to the Protection Profile [**NIAP-OSPP**] and the functional packages [**SSH-FP**, **TLS-FP**], including to the Security Assurance Requirements. As the Protection Profile and the functional packages do not provide an SAR rationale, this ST does not require one.

Anyway, the TOE satisfies the identified assurance requirements as shown on Table 7-1.

SAR Component	How the SAR will be met
ADV_FSP.1	The FINX Team will provide the document "Software Requirements Specification for the FIN.X RTOS SE V5.0". This document describes the external interfaces of the TOE; such as the means for a user to invoke a service and the corresponding response of those services. The description includes: • The interface(s) that enforces a SFR, or supports the enforcement of a SFR; • The interface(s) that does not enforce any SFRs.
AGD_OPE.1, AGD_PRE.1	The FINX Team will provide the document "Software User Manual for the FIN.X RTOS SE V5.0". This document: • Describes the installation, generation, and startup procedures for placing the TOE in the evaluated configuration; • Provides the descriptions of the processes and procedures of how the administrative users of the TOE can securely administer the TOE.
ALC_CMC.1, ALC_CMS.1, ALC_TSU_EXT.1	The FINX Team will provide the documents 1. "Software Configuration Management Plan for the FIN.X RTOS SE V5.0". This document identifies all TOE configuration items, how those configuration items are uniquely identified, and the adequacy of the procedures that are used to control and track changes that are made to the TOE. This includes details on what changes are tracked and how potential changes are incorporated. 2. "Flaws Remediation for the FIN.X RTOS Product Line". This document describes the vulnerability management process for the TOE. 3. "Software Version Document for the FIN.X RTOS SE V5".
ATE_IND.1	The FINX Team will provide the TOE for testing.
AVA_VAN.1	The FINX Team will provide the TOE for testing.

Table 7-1: TOE Security Assurance Measures

8 TOE Summary Specification

This chapter describes the CSCI_FINXSE_V5 security functions that satisfy the security functional requirements of the protection profile and applicable functional packages. A mapping to the TOE security functions to the SFRs is provided.

8.1 TOE Security Functionalities

This section presents the TOE Security Functions (TSFs) and a mapping of security functions to Security Functional Requirements (SFRs). The TOE performs the following security functions:

- **Audit**
- **Cryptographic Support**
- **Identification and Authentication**
- **User Data Protection**
- **Security Management**
- **Protection of the TSF**
- **Trusted Channel/Path**

8.1.1 Audit

8.1.1.1 Audit Generation and Review

The Lightweight Audit Framework (LAF) is an audit system 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 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.

8.1.1.2 Audit Generation and Processing

The kernel interface that provides the means to configure the audit properties is usable only by authorized administrators. Only processes running with *root* privileges or kernel functions can submit audit records. The events to be audited are then forwarded by the kernel to an audit daemon, which 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, or it is halted, or the audit daemon executes an administrator-specified notification action depending on the configuration of the audit daemon, or it may just drop the audit records.

Access to audit data by normal users is prohibited by the discretionary access control mechanism of the TOE, which is used to restrict the access to the audit trail and audit configuration files to the authorized administrator only.

The authorized administrator can define the events to be audited (as detailed by FAU_GEN.1.1) from the overall events that the Lightweight Audit Framework provides, 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 authorized 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 authorized administrator can select files to be audited by adding them to a watch list that is loaded into the kernel.

8.1.1.3 Audit Review

An audit record consists of one or more lines of text containing fields in a “keyword=value” tagged format. At least the following information is contained in all audit record lines:

- Type: indicates the source of the event (e.g., SYSCALL, FS_WATCH, USER, or LOGIN).
- Timestamp: Date and time the audit record was generated.
- Event (audit) ID: unique numerical event identifier.
- Login ID (“auid”), the user ID of the user authenticated by the TOE (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)

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 the authorized administrator only.

8.1.1.4 SFR Summary

- FAU_GEN.1: The TOE generates and manages audit records;
- FMT_MOF_EXT.1, FMT_SMF_EXT.1: The TOE allows the administrator only to configure the LAF service and to access audit log files.

8.1.2 Cryptographic Support

The TOE implements different cryptographic algorithm suites via the OpenSSL software package for supporting utilities that implements protocols for protecting the integrity and confidentiality of both data-at-rest and data-in-motion.

Specifically, the following cryptographic algorithm suites are provided:

- Encryption and decryption (AES), whose specification demanded for this evaluation are defined by [**FIPS197**] and NIST SP 800-38A/38E/38D,
- Hashing (SHA-1, SHA-2), whose specification demanded for this evaluation are defined by FIPS180-4,

- Hashed message authentication code (HMAC), whose specification demanded for this evaluation are defined by FIPS198-1,
- Digital signatures (RSA, DSA, ECDSA), whose specification demanded for this evaluation are defined by NIST SP 800-131A and FIPS186-4,
- Key establishment, whose specification demanded for this evaluation are defined by NIST SP 800-56A/56B,
- Deterministic random bit generation (DRBG), whose specification demanded for this evaluation are defined by NIST SP 800-90A and NIST SP 800-57;
- Zeroization of both volatile and non-volatile memory containing security parameters.

When directly generated via the *openssl* application, asymmetric keys may be encrypted with the AES cipher. If encryption is used, a pass phrase will be prompted for if it is not supplied via the *-passout* argument. The following policy applies to the passphrase by default:

- ✓ Passphrase is 4 to 1024 characters in length.

When directly generated via the *ssh-keygen* application, asymmetric keys may be protected by a passphrase of arbitrary length.

The *OPENSSL_cleanse()* OpenSSL's API performs zeroization of volatile memory while zeroization of volatile memory is performed via the *scrub* utility and enforced by the *trim* utility (if the hardware supports TRIM).

8.1.2.1 Secured network communication channels

8.1.2.1.1 TLS Protocol

The TOE implements the TLS v1.2 protocol and no earlier TLS version by means of the APIs provided by the OpenSSL software package, which support all cipher suite listed in FCS_TLSC_EXT.1 and FCS_TLSS_EXT.1 . The TLS v1.2 protocol enables a trusted network path that is used for client and server communication. Table 8-1 summarizes the TLS RFCs implemented by the TOE:

RFC #	Name	How Used
5246	The Transport Layer Security (TLS) Protocol Version 1.2	Obsoletes RFCs 3268, 4346, and 4366. Specifies requirements for TLS 1.2, including mutual authentication via x.509 certificates.
5280	Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile	It profiles the X.509 v3 certificate and X.509 v2 certificate revocation list (CRL).
5288	AES Galois Counter Mode (GCM) Cipher Suites for TLS	Describes the use of the AES-GCM as a Transport Layer Security (TLS) authenticated encryption operation
5289	TLS Elliptic Curve Cipher Suites with SHA-256/384 and AES Galois Counter Mode (GCM)	Describes elliptic curve cipher suites for Transport Layer Security (TLS)
6125	Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)	Internet Public Key Infrastructure Using X.509 (PKIX) certificates in the context of Transport Layer Security (TLS)

Table 8-1: TLS RFCs Implemented by the TOE

The TOE supports the generation of the RSA and ECDSA key pair and certificates via the *openssl* utility. The key generation mechanism uses the OpenSSL random number generator, which is seeded by the *getrandom* system call. 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). The OpenSSL library provides the functions to set up such a CA, but those functions are not subject of this Security Target.

When negotiating a TLS 1.2 elliptic curve cipher suite, the TOE will include automatically as part of the Client Hello message both its supported elliptic curves extension, i.e., secp256r1, secp384r1, and secp521r1 as well as signature algorithm, i.e., SHA256, SHA384, and SHA512 based on the ciphersuites selected by the administrator.

By default, the signature algorithms in the Client Hello message are: SHA256, SHA384 and SHA512, which meet the FCS_TLSC_EXT.3 requirement. Any service using the TLS will check that the identifying information in the certificate matches what is expected, including the expected Distinguished Name (DN), Subject Name (SN), or Subject Alternative Name (SAN) attributes along with any applicable extended key usage identifiers. In case of unexpected mismatching, a TLS connection is rejected.

All default TLS services enforce mutual authentication. The authentication of the TLS server certificate is performed as follows:

- Using FQDN: When the server's FQDN can be resolved, the TOE tries to match its name with either the CN part of the DN or with the DNS Name or URI Name entry in the SAN. If a CN and a SAN are present, the SAN takes precedence.
- Using IP address: When the server's FQDN cannot be resolved, the TOE tries to match its IP address with either the CN part of the DN or with the IP entry in the SAN. If a CN and a SAN are present, the SAN takes precedence.

The TOE supports wild cards for the server identifier resolution as well as the TLS supported groups extension in the client hello without specific configurations.

Certificate pinning is not supported by the TOE.

8.1.2.1.2 SSH Protocol

The TOE implements SSH v2.0 protocol (RFC 4151 and related RFCs. See Table 8-2) by means of the OpenSSH software package. 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. The cryptographic implementations ensure that sensitive data is appropriately zeroized before releasing the associated memory.

The following security functions are provided:

- Establish a secure communication channel.
  - Encryption and decryption, as defined by RFC 4253 and RFC 4344. When AES-CTR is used, the counter value will be derived from the shared secret obtained during the SSH handshake like the symmetric key. The counter is incremented by one and is a 32-bit value. Due to the maximum life time of a key, the counter value can never overflow;
  - Diffie-Hellman key agreement, as defined RFC 5656 and RFC 8268;

  - Derived session keys, as defined in Section 7.2 of of RFC 4253 when the key establishment scheme uses finite field cryptography (FFC), or as defined in Section 4 of of RFC 5656 when the key establishment scheme uses elliptic curve cryptography (ECC);
  - Key hash for integrity protection, as defined in Section 4.4 of RFC 4253.

  Note: The SSH v2.0 supports some cryptographic algorithms that are not covered by this evaluation and they are not used when running the evaluated configuration of the TOE.

- Perform user authentication using password authentication method, as defined in Section 5 of RFC 4252;
- Perform user authentication using RSA public-key authentication method, as defined in Section 5 of RFC 4252, Section 6.6 of RFC 4253, and Section 3 of RFC 8332;
- Perform user authentication using ECDSA public-key authentication method, as defined in Section 3 of RFC 5656;
- Perform user keyboard-intercative authentication method, as defined in Section 3 of RFC 4256. Multifactor mechanisms may also be configured as:
  - public-key and password authentication methods. Or,
  - public-key and keyboard-intercative authentication methods.
- Check the integrity of the messages exchanged, as defined by RFC 6668.

For each SSH session, the TOE counts the received bytes of the SSH packets. Should any SSH packet be greater than 261,888 bytes, that packet will be dropped. After 1 hour of SSH time connection or after processing SSH packets that total 1 GB of data (excluding any dropped packet), the TOE initiates a re-keying.

Table 8-2 summarizes the SSH RFCs implemented by the TOE.

RFC #	Name	How Used
4251	The Secure Shell (SSH) Protocol Architecture	Describes the architecture of the SSH protocol, as well as the notation and terminology used in SSH protocol documents.
4252	The Secure Shell (SSH) Authentication Protocol	Describes the SSH authentication protocol framework and public key, password, and host-based client authentication methods.
4253	The Secure Shell (SSH) Transport Layer Protocol	Describes the SSH transport layer protocol, which typically runs on top of TCP/IP.
4254	The Secure Shell (SSH) Transport Layer Protocol	Describes the SSH Connection Protocol. It provides interactive login sessions, remote execution of commands, forwarded TCP/IP connections, and forwarded X11 connections. All of these channels are multiplexed into a single encrypted tunnel.
4256	Generic Message Exchange Authentication for the Secure Shell Protocol (SSH)	Describes a general purpose authentication method for the SSH protocol, suitable for interactive authentications where the authentication data should be entered via a keyboard (or equivalent alphanumeric input device).
4344	The Secure Shell (SSH) Transport Layer Encryption Modes	Desbibes solutions for several security problems with the symmetric portion of the SSH Transport Protocol.
5647	AES Galois Counter Mode for the Secure Shell Transport Layer Protocol	Shows how the AES Galois Counter Mode can be used to provide both confidentiality and data integrity to the SSH Transport Layer Protocol.
5656	Elliptic Curve Algorithm Integration in the Secure Shell Transport Layer	Describes algorithms based on Elliptic Curve Cryptography (ECC) for use within the Secure Shell (SSH) transport protocol.

RFC #	Name	How Used
6668	SHA-2 Data Integrity Verification for the Secure Shell (SSH) Transport Layer Protocol	Defines algorithm names and parameters for use in some of the SHA-2 family of secure hash algorithms for data integrity verification in the Secure Shell (SSH) protocol.
8268	More Modular Exponentiation (MODP) Diffie-Hellman (DH) Key Exchange (KEX) Groups for Secure Shell (SSH)	Defines added Modular Exponentiation (MODP) groups for the Secure Shell (SSH) protocol using SHA-2 hashes.
8332	Use of RSA Keys with SHA-256 and SHA-512 in the Secure Shell (SSH) Protocol	Updates RFCs 4252 and 4253 to define new public key algorithms for use of RSA keys with SHA-256 and SHA-512 for server and client authentication in SSH connections.

Table 8-2: SSH RFCs Implemented by the TOE

8.1.2.2 Confidentiality protected data storage

File system objects are stored on block devices, such as partitions on hard disk. The TOE 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. Such additional layer is implemented at kernel level by the *dm_crypt* module which uses the Linux device mapper to provide encryption and decryption operations that are transparent to the file system and therefore to the user.

The block device interfacing with the *dm_crypt* module is an encrypted block device that comply with the LUKS standard [**LUKS**]; for such reason, this kind of block device are hereafter called a 'LUKS-formatted' device.

To be the LUKS-formatted device mounted and then accessed, the owner of the device has to provide a passphrase. This passphrase is used to decrypt the symmetric volume key which is injected into the kernel so enabling it to decrypt (to unlock) the data on the device and to provide access to data stored on that device. Finally, the LUKS-formatted device can be mounted and accessed.

Complexity criterias for passphrases for LUKS-formatted devices can be tuned by means of the */etc/security/pwquality.conf* configuration file. The following complexity criteria are set by default:

- ✓ Passphrase is at least 16 characters in length, consisting of at least two upper and two lower case letters, at least two numbers, and two symbols;
- ✓ Passphrase does not contain more than two equal numbers or letters or symbol consecutives.

Once the LUKS-formatted device is unlocked and mounted, it is accessible as any other file system and alla data are protected by DAC access control. When it is unmounted and locked (i.e. the kernel has not a key to open and access the device), all data on the device is inaccessible, even for privileged users. This protection is active also off-line.

The encryption and decryption operation of the device is implemented by the kernel. The user-level application that allows authorized users to manage LUKS-formatted devices is *cryptsetup*. To enable users to unlock and then access the encrypted device, the *cryptsetup* application performs the following steps:

1. Obtain the user's passphrase and then apply the LUKS key derivation mechanism on it;
2. Extract the encrypted volume key from the device, decrypt it with the key derived from the user's passphrase, and inject the decrypted volume key into the kernel;
3. Set up the mapping between the block device and the volume key.

The *cryptsetup* application uses the OpenSSL random number generator seeded by the *getrandom* system call to generate cryptographic keys. The cryptographic implementations ensure that sensitive data is appropriately zeroized before releasing the associated memory.

Once a volume key has been zeroized, the user owning the passphrase of the affected encrypted volume key is not able to unlock the block device any more.

The *cryptsetup* application provides file encryption services using AES-128 or AES-256 in CBC or XTS mode. The TOE stores all sensitive data (i.e. cryptographic keys for services (stored into */etc/ssl/pki/*), user credentials (see Section 8.1.3)) into the /etc directory. The /etc directory is protected by means of a LUKS-formatted device mounted at boot time.

8.1.2.3 Cryptographic key destruction

Ephemeral cryptographic key materials are held in RAM and used by their process owner. These keys are protected at runtime by process isolation's mechanism, and they are zeroized as soon as they are not required any more by overwriting the memory location with zeros.

The following keys are managed by OpenSSL on behalf of applications that implement the TLS and SSH protocols:

- Ephemeral symmetric keys and ephemeral MAC keys: they are generated via the DH key operation during the TLS/SSH handshake and stay in volatile memory until rekey or destruction of the connection;
- Ephemeral DH/ECDH keys: they are generated using a DRBG during the TLS/SSH handshake from the Diffie-Hellman operation and stay in volatile memory until the handshake is completed;
- Ephemeral representation of asymmetric keys: they locate into files in non-volatile memory. Applications that handle TLS/SSH connections load these keys from file into volatile memory in order to perform the protocol operations. Therefore, these keys stay in volatile memory until needed.

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 volatile memory is zeroized.

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

- Ephemeral symmetric key: after the decrypted volume key is injected into the kernel (see Section 8.1.2.2), memory buffers that contain the passphrase and other uneeded derived keys are overwritten with zeros. When the *dm-crypt* partition is unmounted, the Linux kernel crypto API cipher handle is deallocated which also zeroizes the memory buffer holding the key;
- Master volume key: the [LUKS] standard states that the LUKS header only holds the wrapped master volume key, and therefore it is not subject to zeroization requirements.

The life of asymmetric key (and certificate) files as well as LUKS passphrase files is undefinite. A user or administrator can securely destroy them by means of the *scrub* utility. To erase key material held into a SSD, the TOE provides the *fstrim* utility. After a deletion of a file with sensitive data (via *scrub*), *fstrim* uses the SSD TRIM command to inform the SSD to discard unused blocks bypassing wear leveling.

8.1.2.4 Entropy source used by the TOE

The entropy source of the TOE is the Linux-RNG, which is a pseudo-random number generator that uses hardware events detected by the Linux kernel as noise sources.

Noise sources are captured by an input_pool and passed to an entropy pool that collects and processes that noise, and thus accumulate the entropy, which is used to produce random numbers for the user space interface of */dev/urandom*, */dev/random*, the system call *getrandom* and the in-kernel application-programming interface (API) of *get_random_bytes*.

The term "entropy pool" refers to a memory area holding true random data which is processed with a deterministic input and further put through Conditioning[1]. The output function of an entropy pool is based on the SHA-1 hash.

The Linux-RNG operation can be characterized as follows. After the occurrence of a hardware event, such as an interrupt, the event is awarded an entropy estimation by the Linux-RNG. The event time and the event value are mixed into the entropy pool that has a size of 4096 bits. This entropy pool is called *input_pool*, which is accessible via */dev/urandom* and */dev/random* interfaces:

- Unrestricted generation of random numbers is available with */dev/urandom* and the in-kernel function *get_random_bytes*;
- When accessing */dev/random*, random numbers are only generated when the entropy pool received at least 128 bits of initial entropy. After reaching that threshold of 128 bits of entropy once, */dev/random* will operate non-blocking for the lifetime of the system and thus operate identically to */dev/urandom*.

In addition, the Linux kernel offers the *getrandom* system call for obtaining a series of random bytes. See its man page at *https://man7.org/linux/man-pages/man2/getrandom.2.html* for details.

8.1.2.4.1 Noise sources of the TOE

The noise sources of the TOE can be characterized as follows:

- If specialized hardware random number generators are available, the Liux kernel provides drivers that use the internal *add_hwgenerator_randomness* Kernel function for getting their random data.
- Human Interface Devices (HID) such as keyboard or mice may provide entropy via the *add_input_randomness* Kernel function (invoked by the device driver). The data obtained by HID events such as a pressed key or mouse movement is supplemented with a time stamp that the Linux-RNG obtains when an event arrives using the *add_timer_randomness* Kernel function.

---

[1] "*Conditioning is the process where input data is processed such that the resulting data will not allow an observer to derive the original input data. In addition, conditioning is also the process to reduce statistical weaknesses exhibited in the raw data stream. Such conditioning operations can be performed using cryptographic or non-cryptographic operations. An example for cryptographic conditioning is the application of a hash. A linear feedback shift register (LFSR) is an example for a non-cryptographic conditioning operation*" [BSI - Documentation and Analysis of the Linux Random Number Generator, rev. 4.1].

- Read and write events pertaining to any kind of block devices provide entropy via the *add_disk_randomness* Kernel function. Similarly to the HID noise source, the Linux-RNG adds a time stamp to each disk event by invoking the *add_timer_randomness* Kernel function. Not all block devices will contribute as a noise source. For example, solid-state-drives (SSD) are not used as noise sources whereas hard disks with spinning disks are used as such.
- When an interrupt arrives, the Linux-RNG is triggered with the *add_interrupt_randomness* Kernel function. For each received interrupt, the Linux-RNG obtains a time stamp and supplemental data which is fed into a *fast_pool* instance that is local to the CPU on which the interrupt is processed.
- During boot time when a user space caller requests data from */dev/random* or the *getrandom* system call and the Linux-RNG has not yet obtained 128 bits of entropy, the Linux-RNG tries to generate entropy from the interaction of a high-resolution timer and the Linux kernel scheduler. For this, the kernel first verifies whether it has a high-resolution time stamp. If so, it kicks off the entropy generation logic that runs in parallel with the remainder of the Linux-RNG operation. If the entropy generation fails, the entropy pool will not gain any additional data and the entropy estimator remains unchanged.

The Linux-RNG uses noise sources for implementing a Deterministic Random Bits Generator (DRBG) as recommended by the NIST SP800-90A with the following properties:

- CTR DRBG with AES-128, AES-192, AES-256;
- Hash DRBG with SHA-1, SHA-256, SHA-384, SHA-512;
- With and without prediction resistance.

CTR DRBG with AES-256 is the default setting of the TOE.

8.1.2.4.2 Entropy health test

The TOE implements a entropy health test on system initialization by means of the invocation of the *drbg_healthcheck_sanity* Kernel function, which provide for entropy health test as defined on SP800-90A, sections 11.3.2 and 11.3.3.

Testing of the uninstantiate function, as demanded by SP800-90A, sections 11.3.5, is not required during health testing.

Testing the reseed function, as demanded by SP800-90A, sections 11.3.4, is intended for running on-demand and it is provided by the *drbgtest* utility of the OpenSSL software package.

8.1.2.5 SFR Summary

- FCS_CKM.1, FCS_CKM.2, FCS_COP.1(SYM), FCS_COP.1(HASH), FCS_COP.1(SIGN), FCS_COP.1(HMAC), FCS_RBG_EXT.1: types of keys used by the TOE. .
- FCS_CKM_EXT.4: The TOE overwrites critical cryptographic parameters immediately after that data is no longer needed. Users are provided with a proper utility for zeroizing their private keys.
- FCS_STO_EXT.1: The TOE provides the interface to encrypt and decrypt sensitive data at rest by means of the LUKS standard implemented by the Cryptsetup software package jointly with the Kernel package.

- FCS_TLS_EXT.1, FCS_TLSC_EXT.1, FCS_TLSC_EXT.1(TLS), FCS_TLSC_EXT.2, FCS_TLSC_EXT.2.1(TLS), FCS_TLSC_EXT.3, FCS_TLSC_EXT.4, FCS_TLSC_EXT.5, FCS_TLSS_EXT.1, FCS_TLSS_EXT.2, FCS_TLSS_EXT.3: The TOE provides an TLS v1.2 (or higher) implementation as described about in section 8.1.2.1.1.
- FCS_SSH_EXT.1, FCS_SSHC_EXT.1, FCS_SSHS_EXT.1: The TOE provides an SSH v2.0 implementation as described in section 8.1.2.1.2.

8.1.3 Identification and Authentication

When a user possesses an identity in a system in the form of a login ID and group ID, that user has Identification. Identification establishes user accountability and access restrictions for actions on the TOE. Authentication is verification that the user's claimed identity is valid.

All discretionary access-control decisions made by the kernel are based on the process's user ID established at login time and all mandatory access control decisions made by the kernel are based on the process domain established through login.

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

Authentication on the local console is based on user name and password. Authentication via the SSH protocol first performs the public-key-based authentication which is followed by the user name and password authentication if the public-key-based authentication was unsuccessful.

User names and passwords are stored into an administrative database only accessible to authorized administrators. The TOE offers password quality enforcement mechanisms, which are enforced at the time when the password is changed.

SSH private keys used for public-key-based authentication are accessible to the owner only, by default.

Local user names and passwords credentials are stored in the */etc/passwd* and */etc/shadow* files. The */etc/passwd* file contains usernames, associated IDs, an indicator whether the password of the user is valid, the principal group ID of the user and other (not security relevant) information. The */etc/shadow* file contains a hash of the user's password, the user ID, the time the password was last changed, the expiration time, and the validity period of the password.

In case of LDAP centralised user accounts, user names and passwords credentials are stored in the LDAP database into the */etc/openldap* directory. The LDAP database contains all user data managed for local users in the */etc/passwd* and */etc/shadow* files.

Users are also warned to change their passwords at login time if the password will expire soon and are prevented from logging in if the password has expired. Users can change their own password. Only administrators can add or delete users or change their properties.

Identification and authentication services use a common mechanism for authentication described in this section. This chapter also describes the authentication process for those network services that require authentication.

8.1.3.1 Identification and Authentication mechanisms

Linux uses a suite of libraries called the "Pluggable Authentication Modules" (PAM) that allow an authorized administrator to choose how PAM-aware applications authenticate users. The TOE provides PAM modules that implement the security functionalities that:

- Provides login control and establish identification IDs for a subject (e.g., UID, GID, etc.);
- Ensures the quality of passwords;
- Consults a user's password "history" and not allowing them to re-use olds passwords;
- Enforces limits for accounts;
- Restricts the use of the root account to certain terminals;
- Restricts the use of the *su* command;
- Enforces locking of accounts after failed login attempts.

The login processing sets the real, file system, effective and login UID as well as the real, effective, file system GID and the set of supplemental GIDs of the subject that is created.

During login processing, a banner is shown to the user. The TSF realizes identification and authentication before any action. After successful authentication, the login time is recorded.

After a successful identification and authentication, the TOE initiates a session for the user and spawns the initial login shell as the first process the user can interact with. The TOE provides a mechanism to lock a session upon the user's request.

8.1.3.1.1 SSH public-key-based authentication

In addition to the PAM-based authentication outlined above, the OpenSSH server is able to perform a public-key-based authentication. When a user wants to log on, instead of providing a password, the user applies his SSH key. After a successful verification, the OpenSSH server considers the user as authenticated and performs the PAM-based operations as outlined above.

To establish a public-key-based authentication, a user first has to generate an RSA (or ECDSA) key pair.

The private part of the key pair remains on the client side. The public part is copied to the server into the file *.ssh/authorized_keys* which resides in the home directory of the user he wants to log on as. When the login operation is performed the SSH v2.0 protocol tries to perform the "public-key-based" authentication using the private key on the client side and the public key found on the server side. The operations performed during the public-key-based authentication is defined in RFC 4252 chapter 7.

Should the public-key-based authentication fail, password-based authentication will be used.

8.1.3.2 User Identity and Role Changing

Users can change their identity (i.e., switch to another identity) using the *su* and *sudo* commands.

8.1.3.2.1 su command

The *su* command is intended for a switch to another identity that establishes a new login session and spawns a new shell with the new identity. When invoking *su*, the user must provide the credentials

associated with the target identity - i.e. when the user wants to switch to another user ID, it has to provide the password protecting the account of the target user.

When switching identities, the real, file system and effective user ID and real, file system and effective group ID are changed to the one of the user specified in the command (after successful authentication as this user).

The primary use of the *su* command within the TOE is to allow authorized administrators the ability to assume the "root" administrative identity to perform administrative actions. The use of the *su* command to switch to *root* has been restricted to users belonging to a special group (*admins*). Note that when a user executes a program that has the setuid bit set, only the effective user ID and file system ID are changed to that of the owner of the file containing the program while the real user ID remains that of the caller. The login ID is neither changed by the *su* command nor by executing a program that has the setuid or setgid bit set as it is used for auditing purposes.

The *su* command invokes the PAM authentication mechanism to validate the supplied authentication.

8.1.3.2.2 sudo command

The *sudo* command is intended for giving users permissions to execute commands with another user identity. When invoking *sudo*, the user has to authenticate with this credentials.

sudo is associated with sophisticated ruleset that can be engaged to specify the followings:

- Source user ID;
- Originating from which host;
- Can access a command, a command with specific configuration flags, or all commands within a directory;
- With which new user identity.

The *sudo* command invokes the PAM authentication mechanism to validate the supplied authentication.

8.1.3.3 User Session Lock

The TOE can use the *vlock* application (invoked directly by the user or indirectly by means of the *tmux* application) for locking the current session of the user either after an administrator-specified time of inactivity or upon the user's request.

To unlock the locked session, the user must type in his or her password; then, PAM authentication mechanism is invoked to validate the supplied authentication.

8.1.3.4 Management of Authentication Data

Each TOE instance maintains its own set of users with their passwords and attributes. Although the same human user may have accounts on different deployed systems interconnected by a network and running an instantiation of the TOE, those accounts and their parameter are not synchronized on different TOE instances. Existing mechanism for synchronizing authentication data within the whole distributed deployed system are not subject to this evaluation.

Authorized users are allowed to change their passwords by using the *passwd* command.

Users are prevented from logging in if the password has expired or their account is disabled or suspended.

Only a definite number (by root user or authorized administrator) of login failures are permitted before *login* command exits and:

1. Account for authorized user disabled;
2. Account for authorized administrator suspended (i.e. temporarily disabled) for a definite (by root user or authorized administrator) time period.

At password's expiring time, users are prompted for a new password before proceeding. Users are forced to provide their old password and the new password before continuing, and new passwords are required to satisfy specific complexity criteria.

When SSH public-key-based authentication is in place, users will have to protect their private key part the same way as protecting a password. Appropriate permission settings on the file holding the private key is necessary. To strengthen the protection of the private key, the user can encrypt the key where a password serves as key for the encryption operation.

The time of the last successful logins is recorded in the directory /var/log/tallylog where one file per user is kept.

The TOE displays informative banners before or while users are logging in. The banners are specified also for remote logins (e.g., via SSH).

8.1.3.5 X.509 Certificate Validation

Every TOE component that uses X.509 certificates is responsible for performing certificate validation, however all components validate certificates as described in RFC 5280; both OCSP and CRL services are available to applications for checking the certificate revocation status. Every component that uses X.509 certificates will have a repository for public certificates and will select a certificate based on criteria such as entity name for the communication partner, any extended key usage constraints, and cryptographic algorithms associated with the certificate. The TOE component will use the same kinds of information along with a certification path and certificate trust lists as part of deciding to accept the certificate.

When applying the bi-directional certificate validation for TLS, the TOE can be configured to verify the certificate's distinguished name (DN). The verification of the certificate's DN is performed after the certificate validation part of the bi-directional certificate validation that ensures that the client certificate is trustworthy.

If certificate validation fails, or if the TOE is not able to check the validation status for a certificate, the TOE will not establish a trusted network channel, i.e. the process for establishing the trusted network channel is aborted.

When the TOE needs to generate a certificate enrollment request it must include a distinguished name, information about the cryptographic algorithms used for the request, any certification extensions, and information about the client requesting the certificate.

8.1.3.5.1 Certificate Storage

Stored certificates known as *trusted root certificates* are contained in certificate stores. Each service has its own certificate store; access to a certificate store is managed by the discretionary access control policy.

8.1.3.6 SFR Summary

- FIA_UAU.5: The TOE provides authentication using a username and password;
- FTA_TAB.1: A banner will be displayed prior to allowing a user to logon;
- FIA_AFL.1: After the number of consecutive failed authentication attempts for a user account has been reached, the TOE can be configured to lockout the user account;
- FIA_X509_EXT.1: The TOE validates X.509 certificates according to RFC 5280 and provides OCSP and CRL services for applications to check certificate revocation status;
- FIA_X509_EXT.2, FIA_X509_EXT.1: The TOE uses X.509 certificates for TLS, code signing for system software updates, and code signing for integrity verification;
- FCS_SSH_EXT.1: The TOE uses password-based and public-key-based authentication methods during the authentication steps of SSH sessions;
- FCS_SSHC_EXT.1: The TOE authenticates its peer (SSH server);
- FCS_SSHS_EXT.1: The TOE authenticates its peer (SSH client).

8.1.4 User Data Protection

This section outlines the general DAC policy in the TOE as implemented for resources where access is controlled by permission bits and POSIX ACLs; principally these are the objects in the file system. In all cases the policy is based on user identity (and in some cases on group membership associated with the user identity). To allow for enforcement of the DAC policy, all users must be identified and their identities authenticated.

8.1.4.1 General DAC Policy

The general policy enforced is that subjects (i.e., processes) are allowed only the accesses specified by the policies applicable to the object the subject requests access to. Further, the ability to propagate access permissions is limited to those subjects who have that permission, as determined by the policies applicable to the object the subject requests access to.

A subject with a file system user ID of 0 is exempt from all restrictions of the discretionary access control and can perform any action desired. For the execution of a file by root, the permission bit vector of that file must contain at least one execute bit.

DAC provides the mechanism that allows users to specify and control access to objects that they own. DAC attributes are assigned to objects at creation time and remain in effect until the object is destroyed or the object attributes are changed. DAC attributes exist for, and are particular to, each type of named object known to the TOE. DAC is implemented with permission bits and, when specified, ACLs.

The outlined DAC mechanism applies only to named objects that can be used to store or transmit user data. Other named objects are also covered by the DAC mechanism but may be supplemented by further restrictions. These additional restrictions are out of scope for this evaluation. Examples of objects that are accessible to users but cannot be used to store or transmit user data are: virtual file systems externalizing kernel data structures (such as most of *procfs*, *sysfs*, *binfmt_misc*) and process signals.

During creation of objects, the TSF ensures that all residual contents is removed from that object before making it accessible to the subject requesting the creation.

When data is imported into the TOE (such as when mounting disks created by other trusted systems), the TOE enforces the permission bits and ACLs applied to the file system objects.

8.1.4.2 Permission bits

The TOE supports standard UNIX permission bits to provide one form of DAC for file system objects in all supported file systems. There can be three categories of users: the owning user, users in the owning group, and other users. The three bits in each set indicate the access permissions granted to each user category: one bit for read (r), one for write(w) and one for execute (x). Note that write access to file systems mounted as read only (e.g. CD-ROM) is always rejected. Also, write access to file system objects marked as immutable is always rejected.

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

8.1.4.3 Special Permission

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

- SUID bit: When an executable marked with the SUID bit is executed, the effective UID of the process is changed to the UID of the owner of the file. The SUID bit for file system objects other than files is ignored.
- SGID bit: When an executable marked with the SGID bit is executed, the effective GID of the process is changed to the owning GID of the file. The SGID bit for file system objects other than files is ignored.
- SAVETXT: When a directory is marked with the SAVETXT bit, only the owner of a file system object in that directory can remove it. This bit is commonly used for world-writable directories like */tmp*. Only processes with the CAP_FOWNER capability are able to remove the file system object if their UID is different from the owning UID of the file system object.

8.1.4.4 Access Control Lists

The TOE provides support for POSIX type ACLs to define a fine-grained access control on a user basis. ACLs are supported for all file system objects stored with the following file systems:

- *ext4*;
- *xfs*;
- *tmpfs*.

An ACL entry contains the following information:

- A tag type that specifies the type of the ACL entry,
- A qualifier that specifies an instance of an ACL entry type,
- A permission set that specifies the discretionary access rights for processes identified by the tag type and qualifier.

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

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

8.1.4.5 IPC objects

The TOE implements the following standard types of IPC mechanisms:

- SYSV Shared Memory,
- SYSV and POSIX Message Queues,
- SYSV Semaphores.

UNIX permission bits govern access to the above-mentioned IPC mechanisms. As the IPC objects of UNIX domain socket special files and Named Pipes are represented as filesystem objects, the access control mechanism covering file system objects are applicable to these IPC mechanisms too.

The TOE maintains IPC object types where each process has its own namespace for that object type: sockets - including network sockets. Access to the socket is only possible by the process whose socket namespace contains the socket reference. Setting of permissions for such objects can be handled using file descriptor passing.

8.1.4.6 SFR Summary

- FDP_ACF_EXT.1: The TOE provides a Discretionary Access Control policy to limit modification and reading of objects by non-authorized users.

8.1.5 Security Management

The security management facilities provided by the TOE are usable by authorized administrators to modify the configuration of TSF. The configuration of TSF is hosted in the following locations:

- Configuration files (or TSF databases)
- Data structures maintained by the kernel and within the kernel memory

The TOE provides applications to authorized administrators to perform various administrative tasks. These applications are documented as part of the administrator and user guidance. These applications are either used to modify configuration files or to access parameters controlled and enforced by the kernel via kernel-provided interfaces to user space. Configuration options are stored in different configuration files. These files are protected using the DAC mechanisms against unauthorized access. It is the task of the persons responsible for setting up and administrating the TOE to ensure that the access control features of the TOE are used throughout the lifetime of the system to protect those databases.

These configuration files are accessed using applications which are able to interpret the contents of these configuration files. Each TOE instance maintains its own TSF database. Synchronizing those databases is

not performed in the evaluated configuration. If such synchronization is required by an organization, it will be the responsibility of an authorized administrator of the TOE to achieve this either manually or with some automated assistance.

To access data structures maintained by the kernel, applications use the kernel-provided interfaces, such as system calls, virtual file systems, netlink sockets, and device files. These kernel interfaces are restricted to authorized administrators or authorized non-administrative users, if applicable, by either using DAC (for virtual file system objects) or special kernel-internal verification checks for each interface.

Security management also comprises the management of a reliable time stamps. Such time stamps are essential for correct time information within audit records.

All management activities are restricted to the root user. Administrative users assigned to the “admins” group are eligible to switch to the root user to perform administrative actions using the sudo application. Therefore, those users are defined to be administrators. This covers all management functions for the mechanisms specified in this chapter.

The following listing enumerates the directories containing security relevant data:

- */etc*: System-wide configuration files are stored here;
- */etc/group, /etc/passwd, /etc/shadow*: System-wide credetials for local users and groups;
- */etc/openldap/*: System-wide credetials for LDAP-managed users and groups;
- */lib*, */lib64*, */usr/lib* and */usr/lib64* contains shared libraries;
- */lib/modules*: Kernel modules and device drivers are located in this director;
- */var/security_events/<hostname>/audit*: Audit data is stored in this directory;
- */var/security_events/<hostname>/syslog-ng*: Syslog data is stored in this directory;
- */bin*, */sbin*, */usr/bin*, and */usr/sbin*, */usr/libexec*, */usr/local/sbin*, */usr/local/bin* contains executables;
- */dev* contains all device-related interfaces.

8.1.5.1 Privileges

Privileges to perform administrative actions are maintained by the TOE. These privileges are separated into privileges to act on data or access functionality in user space and in kernel space.

Functionality accessible in user space are applications that can be invoked by users. In addition, data accessible in user space is either data maintained with an application or data stored in persistent or transient storage objects. Privileges are controlled by permissions to invoke applications and to access data.

Functionality and data maintained by the kernel must be accessed using system calls. The kernel implements a privilege check for functions and data that shall not be accessible by normal users. These privileges are controlled with capabilities that can be assigned to processes. If a process is assigned with a capability, it is allowed to request special operations that other processes cannot. To implement consistency with the UNIX legacy, processes with the effective UID of zero are implicitly given all capabilities. However, these processes may decide to drop capabilities. A subject may possess one or more of the following capabilities, which provide the following exemptions from the DAC mechanism:

- CAP_DAC_OVERRIDE: A process with this capability is exempt from all restrictions of the discretionary access control and can perform any action desired. For the execution of a file, the permission bit vector of that file must contain at least one execute bit;
- CAP_DAC_READ_SEARCH: A process with this capability overrides all DAC restrictions regarding read and search on files and directories;
- CAP_CHOWN: A process with this capability is allowed to make arbitrary changes to a file's UID or GID;
- CAP_FOWNER: Setting permissions and ownership on objects even if the process' UID does not match the UID of the object;
- CAP_FSETID: Don't clear SUID and SGID permission bits when a file is modified;
- 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.

The Linux kernel implements many more capabilities than the above listed capabilities. These unmentioned capabilities protect functions that do not directly cover SFR functionality but need to be protected to ensure the integrity of the system and its resources.

8.1.5.2 Security Management Functions

Table 8-3 maps which activities can be done by an authorized user or authorized administrator. A checkmark indicates which entity can invoke the management function. Standard users, or programs running on their behalf, are not able to modify policy or configuration that is set by the administrator, the result is that the user cannot override the configuration specified by the administrator.

Management Function	Administrator	User
Enable/disable screen lock	√	
Enable/disable screen lock timeout	√	
Configure screen lock inactivity timeout	√	
Configure local audit storage capacity	√	
Configure minimum password Length	√	
Configure minimum number of special characters in password	√	
Configure minimum number of numeric characters in password	√	
Configure minimum number of uppercase characters in password	√	
Configure minimum number of lowercase characters in password	√	
Configure lockout policy for unsuccessful authentication attempts through: • Timeouts between attempts; • Limiting number of attempts during a time period.	√	
Configure host-based firewall	√	
Configure name/address of directory server with which to bind	√	

Configure name/address of remote management server from which to receive management settings	√	
Configure name/address of audit/logging server to which to send audit/logging records	√	
Configure audit rules	√	
Configure name/address of network time server	√	
Enable/disable automatic software update		
Configure WiFi interface		
Enable/disable Bluetooth interface		
Enable/disable network interface cards	√	
Enable/disable USB interfaces	√	
Enable/disable no other external interfaces		
Configure and manage user accounts	√	
No other functions		

Table 8-3: General Purpose OS Security Management Functions

8.1.5.3 SFR Summary

- FPT_ACF_EXT.1: The TOE provides a Discretionary Access Control policy to limit modification and reading of objects by non-authorized users;
- FMT_MOF_EXT.1, FMT_SMF_EXT.1: The TOE provides the user with the capability to administer the security functions described in the security target. The mappings to specific functions are described in each applicable section of the TOE Summary Specification.

8.1.6 Protection of the TSF

While in operation, the kernel software and data are protected by the hardware memory protection mechanisms[2]. The memory and process management components of the kernel ensure a user process cannot access kernel storage or storage belonging to other processes.

Non-kernel TSF software and data are protected by DAC and process isolation mechanisms. In the evaluated configuration, the reserved user ID root owns the directories and files that define the TSF configuration. In general, files and directories containing internal TSF data (e.g., configuration files, batch job queues) are also protected from reading by DAC permissions.

The hardware platform where the TOE is installed and the firmware components are required to be physically protected from unauthorized access. The operating system kernel mediates all access to hardware components that are protected from direct access by user process.

[2] Hardware mechanisms are used by the TOE to provide a protected domain for its execution. They include a multistate processor (hardware privileged mechanism), memory segment protection, and virtual memory page protection. The TOE software relies on these hardware mechanisms to implement TSF isolation, non-circumventability, and process address-space separation.

All TSF data is commonly read-only for users based on the DAC permission bits. For sensitive TSF data such as user passwords or private keys, the permission bits do not allow any access by a user. The following listing enumerates the storage location of TSF data:

- The directory */etc* contains all configuration files for system-wide configurations. This includes the location for key material;
- The directory */dev* contains device files to allow interaction between applications and devices;
- The directories */lib*, */lib64*, */usr/lib* and */usr/lib64* contain shared libraries;
- The directory */lib/modules* contains kernel drivers;
- The directory */usr/share* and all directories in */var* except */var/tmp* contains TSF data specific to certain shared libraries and applications;
- The directories */bin*, */sbin*, */usr/sbin*, */usr/bin*, */usr/libexec*, */usr/local/sbin*, */usr/local/bin* contain TSF executables;
- The directories */sys* and */proc* contain kernel interfaces usable by applications;
- The directory */boot* contains the kernel binary and the boot file system (initramfs).

8.1.6.1 Stack Buffer Overflow Protection

All binaries (i.e. the kernel, kernel modules, executables, and shared libraries) are compiled by adding a stack canary and associated verification code during the entry and exit of function frames.

At function entry, the canary is set with a random value generated using a pseudo-random number generator. Then it is checked at function exit: if the canary value changed, a stack overflow occurred and the binary execution will be aborted.

8.1.6.2 Address Space Layout Randomization

The TOE implements Address Space Layout Randomization (ASLR) for all binaries – except for non-Position-Independent-Executable applications and non-Position-Intependent-Code shared libraries – in order to load executable code at unpredictable base addresses.

The base address is generated using a pseudo-random number generator that is seeded by high quality entropy sources, which provides at least 8 random bits for memory mapping.

8.1.6.3 Secure Boot Support

The Unified Extensible Firmware Interface (UEFI) Secure Boot technology ensures that the system firmware checks whether the system boot loader is signed with a cryptographic key authorized by a database of public keys contained in the firmware. With signature verification in the next-stage boot loader and kernel, it is possible to prevent the execution of kernel space code that has not been signed by a trusted key.

A chain of trust is established from the firmware to the signed drivers and kernel modules as follows. The first-stage boot loader, *shim.efi*, is signed by a UEFI private key and authenticated by a public key, signed by a certificate authority (CA), stored in the firmware database. The *shim.efi* contains the TOE public key, “CSCI_FINX Secure Boot (CA key 1)”, which is used to authenticate both the GRUB 2 boot loader,

grubx64.efi, and the CSCI_FINX kernel. The kernel in turn contains public keys to authenticate drivers and modules.

Secure Boot is the boot path validation component of the Unified Extensible Firmware Interface (UEFI) specification. The specification defines:

- A programming interface for cryptographically protected UEFI variables in non-volatile storage;
- How the trusted X.509 root certificates are stored in UEFI variables;
- Validation of UEFI applications like boot loaders and drivers;
- Procedures to revoke known-bad certificates and application hashes.

UEFI Secure Boot does not prevent the installation or removal of second-stage boot loaders, nor require explicit user confirmation of such changes. Signatures are verified during booting, not when the boot loader is installed or updated. Therefore, UEFI Secure Boot does not stop boot path manipulations, it helps in the detection of unauthorized changes. A new boot loader or kernel will work as long as it is signed by a key trusted by the system.

8.1.6.3.1 UEFI Secure Boot Support

The TOE includes support for the UEFI Secure Boot feature, which means that it can be installed and run on systems where UEFI Secure Boot is enabled. On UEFI-based systems with the Secure Boot technology enabled, all drivers that are loaded must be signed with a trusted key, otherwise the system will not accept them. All drivers provided by the TOE are signed by one of its private keys and authenticated by the corresponding public key in the kernel.

GRUB2 module loading is disabled as there is no infrastructure for signing and verification of GRUB 2 modules, which means allowing them to be loaded would constitute execution of untrusted code inside the security perimeter that Secure Boot defines. Instead, the TOE provides a signed GRUB 2 binary that has all the modules supported already included.

8.1.6.4 Trusted Installation

The TOE shall be delivered in the form of a ISO file or self-install DVD-ROM. In both cases, the TOE is digitally signed. The TOE signature and cerificate files required for signature verification MUST be obtained from an out-of-band channel than the TOE itself.

The TOE user manual provides step-by-step instructions for the secure installation of the TOE.

8.1.6.5 Trusted Update

The TOE software is delivered and installed using the Portage Package Manager, which makes use of the Portage Database for performing package building and installation. Software packages are archives of files and contain *ebuilds* metadata to manage these packages.

All TOE installations come with both Portage Package Manager and Portage Database. A RSA digital signature of the Portage Database is also provided. The TOE will verify the signature of the Portage Database before installing any software package. Only if the signature verification is successful, a software package is installed. Otherwise, it is rejected and not installed.

As the Portage database contains the hash (SHA256) of all software package, once its digital signature is verified it is just need to verify the hash of the software package to ensure its integrity and authenticity.

Software updates can be deployed after updating the Portage database. A new Portage database has its signature that must be verified. The TOE user manual provide the means for getting the digital signature of the new Portage database (via out-of-band channel, for instance).

Software updates i.e. new Portage database are provided on a regular basis. Once updated, the new Portage database removes the old one togheter with its signature. This will prevent rollback attacks.

To be alerted that an updated Portage database is available, the authorized administrator can subscribe to the TOE mailing list. This way is detailed in the TOE user manual.

Portage updates are provided on a half-yearly basis along with TOE Security Advisories, which also contain:

- Known vulnerabilities (CVEs) remediated by the new Portage database;
- Software packages updates.

In case of critical vulnerabilities, these updates are provided as soon as remediations become available. The TOE utilizes CVSS 3.0 specification for scoring CVEs: the TOE user manual details how CVE remediations are prioritised for updating the Portage database.

8.1.6.6 SFR Summary

- FPT_ASLR_EXT.1: The TOE randomizes user-mode process address spaces and kernel-mode address space;
- FPT_SBOP_EXT.1: The TOE binaries are compiled with stack overflow protection;
- FPT_TST_EXT.1: The OS verifies the integrity of the bootchain up through the OS kernel;
- FPT_TUD_EXT.1, FPT_TUD_EXT.2: The TOE provides for software updates. The TOE has update mechanisms to deliver software updates and a means for a user to verify the integrity and authenticity of such updates.

8.1.7 Trusted Channel/Path

The TOE provides trusted network channels to communicate with supporting IT infrastructure or applications, like *syslog-ng*, *sshd*, *postix*, and others.

In order to establish a trusted channel, these communications are protected as described above in section 8.1.2.1.

8.1.7.1 SFR Summary

- FTP_ITC_EXT.1, FCS_TLS_EXT.1, FCS_SSH_EXT.1: The TOE provides several trusted network channels that protect data in transit from disclosure, provide data integrity, and endpoint identification that is used by TLS and SSH;
- FTP_TRP.1: The TOE provides a local trusted path service as described in TOE Access and a network-based trusted channel built on the network protocols described in this section.

9 Rationale for Modifications to the Security Functional Requirements

This Security Target includes security functional requirements (SFRs) that can be mapped to SFRs found in the OS protection profile [NIAP-OSPP] and applied functional packages [SSH-FP], [TLS-FP], along with technical decisions that describe additional features and capabilities. The mapping from protection profile SFRs to security target SFRs along with rationale for operations is presented in Table 9-1. SFR operations left incomplete in the protection profile have been completed in this security and are identified within each SFR in section 6.1.

PP or EP	Requirement	ST Requirement	Operation & Rationale
OS	FCS_CKM.1(1)	FCS_CKM.1	Multiple selections which are allowed by the OS protection profile [NIAP-OSPP].
OS	FCS_CKM.2(1)	FCS_CKM.2	Multiple selections which are allowed by the OS protection profile [NIAP-OSPP].
OS	FCS_CKM_EXT.4	FCS_CKM_EXT.4	Multiple selections and assignments which are allowed by the OS protection profile [NIAP-OSPP]. Applies the NIAP Technical Decision 0365.
OS	FCS_COP.1(1)	FCS_COP.1(SYM)	Multiple selections which are allowed by the OS protection profile [NIAP-OSPP].
OS	FCS_COP.1(2)	FCS_COP.1(HASH)	Applies the NIAP Technical Decision 0578.
OS	FCS_COP.1(3)	FCS_COP.1(SIGN)	Multiple selections which are allowed by the OS protection profile [NIAP-OSPP].
OS	FCS_COP.1(4)	FCS_COP.1(HMAC)	An assignment and multiple selections which are allowed by the OS protection profile [NIAP-OSPP].
OS	FCS_RBG_EXT.1	FCS_RBG_EXT.1	Multiple selections which are allowed by the OS protection profile [NIAP-OSPP].
OS	FCS_STO_EXT.1	FCS_STO_EXT.1	Copied from the OS protection profile [NIAP-OSPP] without changes.
OS	FCS_TLSC_EXT.1	FCS_TLSC_EXT.1	Multiple selections which are allowed by the NIAP Technical Decision 0441.
OS	FCS_TLSC_EXT.2	FCS_TLSC_EXT.2	Multiple selections which are allowed by the OS protection profile [NIAP-OSPP].
OS	FDP_ACF_EXT.1	FDP_ACF_EXT.1	Copied from the OS protection profile [NIAP-OSPP] without changes.
OS	FMT_MOF_EXT.1	FMT_MOF_EXT.1	Copied from the OS protection profile [NIAP-OSPP] without changes.
OS	FMT_SMF_EXT.1	FMT_SMF_EXT.1	Multiple selections and assignments which are allowed by the OS protection profile [NIAP-OSPP].
OS	FPT_ACF_EXT.1	FPT_ACF_EXT.1	Two assignments which are allowed by the OS protection profile [NIAP-OSPP].
OS	FPT_ASLR_EXT.1	FPT_ASLR_EXT.1	A selection and an assignment which are allowed by the OS protection profile [NIAP-OSPP].
OS	FPT_SBOP_EXT.1	FPT_SBOP_EXT.1	A selection that is allowed by the OS protection profile [NIAP-OSPP].
OS	FPT_TST_EXT.1	FPT_TST_EXT.1	Applies the NIAP Technical Decision 0493.
OS	FPT_TUD_EXT.1	FPT_TUD_EXT.1	Applies the NIAP Technical Decision 0463.
OS	FPT_TUD_EXT.2	FPT_TUD_EXT.2	Applies the NIAP Technical Decision 0463.
OS	FAU_GEN.1	FAU_GEN.1	A selection and multiple assignments which are allowed by the OS protection profile [NIAP-OSPP].
OS	FIA_AFL.1	FIA_AFL.1	Multiple assignments and multiple selections which are allowed by the OS protection profile [NIAP-OSPP].
OS	FIA_UAU.5	FIA_UAU.5	A selection and an assignment which are allowed by the OS protection profile [NIAP-OSPP].

PP or EP	Requirement	ST Requirement	Operation & Rationale
OS	FIA_X509_EXT.1	FIA_X509_EXT.1	Multiple selections which are allowed by the NIAP Technical Decision 0525.
OS	FIA_X509_EXT.2	FIA_X509_EXT.2	A selection which is allowed by the OS protection profile [NIAP-OSPP].
OS	FTP_ITC_EXT.1	FTP_ITC_EXT.1	Multiple selections and assignments which are allowed by the OS protection profile [NIAP-OSPP].
OS	FTP_TRP.1	FTP_TRP.1	Multiple selections which are allowed by the OS protection profile [NIAP-OSPP].
OS	FTA_TAB.1	FTA_TAB.1	Copied from the OS protection profile [NIAP-OSPP] without changes.
SSH	FCS_SSH_EXT.1	FCS_SSH_EXT.1	Multiple selections which are allowed by the SSH functional package [SSH-FP].
SSH	FCS_SSHC_EXT.1	FCS_SSHC_EXT.1	Multiple selections and an assignment which are allowed by the SSH functional package [SSH-FP].
SSH	FCS_SSHS_EXT.1	FCS_SSHS_EXT.1	Multiple selections and an assignment which are allowed by the SSH functional package [SSH-FP].
TLS	FCS_TLS_EXT.1	FCS_TLS_EXT.1	Multiple selections which are allowed by the TLS functional package [TLS-FP].
TLS	FCS_TLSC_EXT.1	FCS_TLSC_EXT.1(TLS)	Multiple selections which are allowed by the TLS functional package [TLS-FP]. Applies the NIAP Technical Decision 0442.
TLS	FCS_TLSC_EXT.2	FCS_TLSC_EXT.2(TLS)	Copied from the TLS functional package [TLS-FP] without changes.
TLS	FCS_TLSC_EXT.3	FCS_TLSC_EXT.3	Multiple selections which are allowed by the TLS functional package [TLS-FP].
TLS	FCS_TLSC_EXT.4	FCS_TLSC_EXT.4	Copied from the TLS functional package [TLS-FP] without changes.
TLS	FCS_TLSC_EXT.5	FCS_TLSC_EXT.5	Multiple selections which are allowed by the TLS functional package [TLS-FP].
TLS	FCS_TLSS_EXT.1	FCS_TLSS_EXT.1	Multiple selections which are allowed by the TLS functional package [TLS-FP]. Applies the NIAP Technical Decisions 0442, 0588
TLS	FCS_TLSS_EXT.2	FCS_TLSS_EXT.2	Copied from the TLS functional package [TLS-FP] without changes.
TLS	FCS_TLSS_EXT.3	FCS_TLSS_EXT.3	Multiple selections which are allowed by the TLS functional package [TLS-FP].

Table 9-1: Rationale for Operations

10 Rationale for the TOE Summary Specification

This section, in conjunction with section 8 “TOE Summary Specification”, provides evidence that security functions are suitable to meet the TOE security requirements.

Each subsection in section 8 “TOE Summary Specification”, describes a Security Function (SF) of the TOE. Each description is followed with rationale that indicates which requirements are satisfied by aspects of the corresponding SF. The set of security functions work together to satisfy all of the functional requirements. Furthermore, all the security functions are necessary in order for the TSF to provide the required security functionality.

The set of security functions work together to provide all of the security requirements as indicated in Table 10-1. The security functions described in the TOE Summary Specification and listed in the tables below are all necessary for the required security functionality in the TSF.

PP or EP	Security Target SFRs	Audit	Cryptographic Support	Identification and Authentication	User Data Protection	Security Management	Protection of the TSF	Trusted Channel/Path
OS	FCS_CKM.1		√					
OS	FCS_CKM.2		√					
OS	FCS_CKM_EXT.4		√					
OS	FCS_COP.1(SYM)		√					
OS	FCS_COP.1(HASH)		√					
OS	FCS_COP.1(SIGN)		√					
OS	FCS_COP.1(HMAC)		√					
OS	FCS_RBG_EXT.1		√					
OS	FCS_STO_EXT.1		√					
OS	FCS_TLSC_EXT.1		√					
OS	FCS_TLSC_EXT.2		√					
OS	FDP_ACF_EXT.1				√			
OS	FMT_MOF_EXT.1					√		
OS	FMT_SMF_EXT.1					√		
OS	FPT_ACF_EXT.1						√	
OS	FPT_ASLR_EXT.1						√	
OS	FPT_SBOP_EXT.1						√	
OS	FPT_TST_EXT.1						√	
OS	FPT_TUD_EXT.1						√	
OS	FPT_TUD_EXT.2						√	
OS	FAU_GEN.1	√						
OS	FIA_AFL.1			√				
OS	FIA_UAU.5			√				
OS	FIA_X509_EXT.1			√				
OS	FIA_X509_EXT.2			√				
OS	FTP_ITC_EXT.1							√
OS	FTP_TRP.1							√
OS	FTA_TAB.1			√				
SSH	FCS_SSH_EXT.1		√	√				√
SSH	FCS_SSHC_EXT.1		√					√
SSH	FCS_SSHS_EXT.1		√					√
TLS	FCS_TLS_EXT.1		√					√
TLS	FCS_TLSC_EXT.1(TLS)		√					
TLS	FCS_TLSC_EXT.2(TLS)		√					
TLS	FCS_TLSC_EXT.3		√					
TLS	FCS_TLSC_EXT.4		√					
TLS	FCS_TLSC_EXT.5		√					
TLS	FCS_TLSS_EXT.1		√					
TLS	FCS_TLSS_EXT.2		√					
TLS	FCS_TLSS_EXT.3		√					

Table 10-1: Requirement to Security Function Correspondence

11 Abbreviations and References

11.1 ABBREVIATIONS

ACL	Access Control Lists
AES	Advanced Encryption Standard
AH	Authentication Header
ANSI	American National Standards Institute
API	Application Program Interface
ASCII	American Standard Code for Information Interchange
CA	Certification Authority
CBC	Cipher Block Chaining
CC	Common Criteria
CD	Compact Disk
CD-ROM	Compact Disk - Read Only Memory
CPU	Central Processing Unit
CRL	Certificate Revocation List
CSCI	Computer Software Configuration Item
CTR	Cipher Counter mode
CVE	Common Vulnerability Enumeration
DAC	Discretionary Access Control
DH	Diffie-Hellman
DSA	Digital Signature Algorithm
DVD	Digital Versatile Disk
EAL1	Evaluation Assurance Level 1
FIN.X RTOS	Finmeccanica Linux – Real Time Operating System
FIPS	Federal Information Processing Standards
GCC	GNU Compiler Collection
GID	Group Identifier
H&M	Human and Machine users
HMAC	Hashed Message Authentication Code
I/O	Input/Output
ID	Identifier
IP	Internet Protocol
IPC	Inter-Process Communication
IPv4	IP version 4
IPv6	IP version 6
ISAPI	Internet Server API
ISO	International Organization for Standardization

IT	Information & Technology
LAF	Lightweight Audit Framework
LAN	Local Area Network
LUKS	Linux Unified Key Setup
NIST	National Institute of Standards and Technology
OE	Objective for the Environment
OS	Operating System
OSP	Organisational security policies
P/N	Part Number
PAM	Pluggable Authentication Module
POSIX	Portable Operating System Interface for Unix
PP	Protection Profiles
PRNG	Pseudo-Random Number Generation
PUB	Publication
RAM	Random Access Memory
RFC	Request for Comments
RNG	Random Number Generation
ROM	Read Only Memory
RSA	Ron Rivest, Adi Shamir, Leonard Adleman (authors of an algorithm for public-key cryptography)
RTOS	Real Time Operating System
SA	Security Association
SAR	Security Assurance Requirement
S&F	Success & Failure
SE	Security Enhanced
SF	Security Function
SFP	Security Function Policy
SFR	Security Functional Requirement
SHA	Secure Hash Algorithm
SSH	Secure Shell
ST	Security Target
TLS	Transport Layer Security
TOE	Target Of Evaluation
TSC	TOE SCope
TSF	TOE Security Function
UID	User IDentifier
URL	Uniform Resource Locator
USB	Universal Serial Bus

VPN	Virtual Private Network

11.2 REFERENCES

[CC] Common Criteria for Information Technology Security Evaluation, CCMB-2012-09-001 to CCIMB-2012-09-003, Version 3.1 Revision 5, April 2017 (Part 1 to 3)

[FIPS140] FIPS 140-2: Federal Information Processing Standards Publication – Security Requirements for Cryptographic Modules - May 25, 2001 at http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf

[FIPS186-4] FIPS 186-4: Federal Information Processing Standards Publication – Digital Signature Standard (DSS) – June, 2013, http://csrc.nist.gov/publications/fips/fips186-4/fips_186-4.pdf

[FIPS197] Federal Information Processing Standards (FIPS) Publication 197, Announcing the Advanced Encryption Standard (AES), U.S. DoC/NIST, Nov. 26, 2001, https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.197.pdf

[Gentoo] Gentoo Linux Operating System at http://www.gentoo.org

[NIAP-OSPP] Operating System Protection Profile OSPP v4.2.1 at https://www.niap-ccevs.org/MMO/PP/PP_OS_V4.2.1.pdf

[OMSecPol] OpenSSL v1.1.1 updated with FIPS patches

[PBKDF2] RFC 2898: Password-Based Cryptography Specification Version 2.0 at http://www.ietf.org/rfc/rfc2898.txt

[PREEMPT_RT]Real-time Pre-emption patch for the Linux kernel at http://rt.wiki.kernel.org

[SSH-4251] RFC 4251: The Secure Shell (SSH) Protocol Architecture at http://www.ietf.org/rfc/rfc4251.txt

[SSH-4252] RFC 4252: The Secure Shell (SSH) Authentication Protocol, http://www.ietf.org/rfc/rfc4252.txt

[SSH-4253] RFC 4253: The Secure Shell (SSH) Transport Layer Protocol, http://www.ietf.org/rfc/rfc4253.txt

[SSH-4254] RFC 4254: The Secure Shell (SSH) Transport Layer Protocol, https://tools.ietf.org/rfc/rfc4254.txt

[SSH-4256] Generic Message Exchange Authentication for the Secure Shell Protocol (SSH), https://www.ietf.org/rfc/rfc4256.txt

[SSH-5647] RFC 5647: AES Galois Counter Mode for the Secure Shell Transport Layer Protocol, http://www.ietf.org/rfc/rfc5647.txt

[SSH-5656] RFC 5656: Elliptic Curve Algorithm Integration in the Secure Shell Transport Layer, http://www.ietf.org/rfc/rfc5656.txt

[SSH-6668] RFC 6668: SHA-2 Data Integrity Verification for the Secure Shell (SSH) Transport Layer Protocol, https://www.ietf.org/rfc/rfc6668.txt

[SSH-8268] More Modular Exponentiation (MODP) Diffie-Hellman (DH) Key Exchange (KEX) Groups for Secure Shell (SSH), https://datatracker.ietf.org/doc/html/rfc8268

[SSH-8332] Use of RSA Keys with SHA-256 and SHA-512 in the Secure Shell (SSH) Protocol, https://tools.ietf.org/html/rfc8332

[SSH-FP] Functional Package for Secure Shell (SSH), Version 1.0, 2021-05-13at https://www.niap-ccevs.org/MMO/PP/ pkg_ssh_v1.0.pdf

[TLS-5246] The Transport Layer Security (TLS) Protocol Version 1.2, https://tools.ietf.org/html/rfc5246

[TLS-5280] Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, https://datatracker.ietf.org/doc/html/rfc5280

[TLS-5288] AES Galois Counter Mode (GCM) Cipher Suites for TLS, https://tools.ietf.org/html/rfc5288

[TLS-5289] TLS Elliptic Curve Cipher Suites with SHA-256/384 and AES Galois Counter Mode (GCM), https://tools.ietf.org/html/rfc5289

[TLS-6125] Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS), https://tools.ietf.org/html/rfc6125

[TLS-FP] Functional Package for Transport Layer Security 1.1, 2019-02-12 at https://www.niap-ccevs.org/MMO/PP/PKG_TLS_V1.1.pdf

[LUKS] Linux Unified Key Setup, V1.1.1, December 2010