FIN.X RTOS SE V4.0

Security Target

Revision History

Rev.	Change Order	Date	Changes Description
01	First Release	29.01.2015	--
02	PR: 16A00005921,A,1 CP: 16B00006242,A,1	26.06.2015	1. General: changed TSF to TOE (where more suitable) 2. General: changed FINX RTOS to TOE (where more suitable) 3. General: referenced support for ext2, ext3, and ext4, filesystem 4. Section 1 – minor changes 5. Section 1.3 – added “TSF data” and “User data” definition 6. Section 1.5.3 – on the hardware used during the evaluation process: added board VPB14/033 and changed board GEXVR15 to GEXVR16 7. Section 1.6 – minor changes 8. Section 1.6.3.6 – updated description of cryptographic services 9. Section 3.1 – minor changes 10. Section 3.1.3 – added new threats countered by the TOE: T.TSFFUNC, T.USER, T.TSFDATA, and T.USERDATA 11. Section 3.2 – minor changes 12. Section 3.2 – updated policy to restrict allowed ciphers to FIPS-allowed ones 13. Section 3.2 – better description for security objectives. Added the O.RUNTIME.PROTECTION security objective 14. Section 4.3.1 – Table 1: updated correlation between threats and security objectives 15. Section 4.3.2 – Table 3: provided justification that the new/updated security objectives are suitable to counter all the threats 16. Section 5 – added the Extended Functional Requirements FCS_CKM_EXT 17. Section 5 – deleted the Extended Functional Requirements FIA_AFL_EXT.1 18. General: changed SFR FIA_AFL_EXT.1 to FIA_AFL.1 19. Section 6.1.5 – updated SFR FAU_SEL.1.1 to include system call number attribute 20. Section 6.1.8 – updated SFR FAU_STG.3 to include “size” limit for the audit trail 21. Section 6.2.1: ○ Updated SFR FCS_CKM.1.1 to restrict allowed ciphers to FIPS-allowed ones ○ Renamed FCS_CKM.1.1 to FCS_CKM.1(1).1 22. Sections 6.2.2 – renamed FCS_CKM.1.1 to FCS_CKM.1(2).1 23. Sections 6.2.3 ○ Updated SFR FCS_CKM.1.1 ○ Renamed FCS_CKM.1.1 to FCS_CKM.1(3).1 24. Sections 6.2.4 ○ Updated SFR FCS_CKM.1.1 ○ Renamed FCS_CKM.1.1 to FCS_CKM.1(4).1 25. Section 6.2.5 – updated SFR FCS_CKM.2.1 for it to reference RFC4253 26. Section 6.2.6 – deleted SFR FCS_CKM.4.1 27. Section 6.2.6 – updated SFR FCS_COP.1.1 for encryption, decryption, integrity verification, and peer authentication 28. Section 6.2.7

			○ Updated SFR FCS_COP.1.1 for encryption, decryption, and integrity verification ○ Renamed FCS_COP.1.1 to FCS_COP.1(1).1 29. Section 6.2.8 – renamed FCS_COP.1.1 to FCS_COP.1(2).1 30. Section 6.2.9 – deleted FCS_RNG_EXT.1 31. Section 6.3.2 – updated SFR FDP_ACF.1.1 and FDP_ACF.1.2 to include ext2, ext3, and ext4, filesystem 32. Section 6.4.1 – renamed SFR FIA_AFL_EXT.1 to FIA_AFL.1 33. Section 6.4.2 – typo on SFR FIA_ATD.1.1: "unique" intead of "user" 34. Section 6.4.3 – updated SFR FIA_SOS.1.1 to include public-key-based authentication methods 35. Section 6.4.5 – added SFR FIA_UAU.5.1 to define the allowed authentication mechanisms 36. Section 6.4.5 – added SFR FIA_UAU.5.2 to define the authentication rules 37. Section 6.5.1 – renamed FMT_MOF.1.1 to FMT_MOF.1(1).1 38. Section 6.5.2 – renamed FMT_MOF.1.1 to FMT_MOF.1(2).1 39. Section 6.5.6 – renamed FMT_MTD.1.1 to FMT_MTD.1(1).1 40. Section 6.5.7 – renamed FMT_MTD.1.1 to FMT_MTD.1(2).1 41. Section 6.5.8 – renamed FMT_MTD.1.1 to FMT_MTD.1(3).1 42. Section 6.5.9 – renamed FMT_MTD.1.1 to FMT_MTD.1(4).1 43. Section 6.5.10 – renamed FMT_MTD.1.1 to FMT_MTD.1(5).1 44. Section 6.5.11 – renamed FMT_MTD.1.1 to FMT_MTD.1(6).1 45. Section 6.5.12 – renamed FMT_MTD.1.1 to FMT_MTD.1(7).1 46. Section 6.5.13 – renamed FMT_REV.1.1 to FMT_REV.1(1).1 47. Section 6.5.14 – renamed FMT_REV.1.1 to FMT_REV.1(2).1 48. Section 6.5.15 – updated SFR FMT_SAE.1.2 to better define user's reference 49. Section 6.6.3: ○ Added SFR FPT_FLS.1.1 to preserve a secure state of the TOE: full buffer overflow protection ○ Renamed FPT_FLS.1.1 to FPT_FLS.1(1).1 50. Section 6.6.4: ○ Added SFR FPT_FLS.1.1 to preserve a secure state of the TOE: partial buffer overflow protection ○ Renamed FPT_FLS.1.1 to FPT_FLS.1(2).1 51. Section 6.7.1 – updated SFR FTA_SSL.2.2 to restrict applicability on password-based authentication method 52. Section 6.7.3 – updated SFR FTA_TAH.1.1 and FTA_TAH.1.2 to include the "location"information for access history 53. Section 6.8.1 – updated SFR FTP_ITC.1.1: word "unauthorized" was deleted 54. Section 6.8.1 – updated SFR FTP_ITC.1.2: "the TSF or the remote" was replaced with "another" 55. Section 6.9.1 – updated Table 8 with new/deleted SFRs and security objectives 56. Section 6.9.2 – updated Table 9 with new security objectives 57. Section 6.9.3 – updated Table 10 with new/deleted SFRs 58. Section 7.1 – minor changes 59. Section 8.1 – minor changes 60. Section 8.1.1.1.1 – new section for SSH public-key-based authentication 61. Section 8.1.1.3 – changed "screen" to "vlock" 62. Section 8.1.1.4 – mentioned /var/log/tallylog, where one file per

			user is kept 63. Section 8.1.3.3 – added ext2, and ext4, filesystem 64. Section 8.1.5 – minor changes 65. Section 8.1.6 – re-defined the cryptographic services 66. Section 8.1.7.1 – minor changes 67. Section 8.1.7.8 – new section for describing protection mechanisms against buffer overrun vulnerability
03	CO: 16C00006466,A,1	03.04.2017	1. General: removed support for ext2 filesystem 2. General: • renamed FCS_CKM_EXT.4 to FCS_CKM.4_EXT • renamed FCS_CKM_EXT.4.1 to FCS_CKM.4.1_EXT 3. General: • renamed FPT_FLS.1(FULL) to FPT_FLS.1(1) • renamed FPT_FLS.1(PARTIAL) to FPT_FLS.1(2) 4. Section 1.5.3: • removed the "GEXVR16 C-Model, http://defense.ge-ip.com/products/xvr16/p3636" from the hardware used during the evaluation process • added USB ports 5. Section 1.6.2: minor changes 6. Section 1.6.3.1: provided a better description for the provided authentication mechanisms 7. Section 3.2: modified the P.PWD_QUALITY 8. Section 4.1: modified the O.USER_AUTHENTICATION 9. Section 4.3.2: removed redoundant row (last three) from Table 5 10. Section 5.1.1: removed subsections 5.1.1.1, 5.1.1.2, 5.1.1.3, and 5.1.1.6 11. Section 6.1.1: updated table 7 12. Section 6.2.1: updated application note for FCS_CKM.1(1) 13. Section 6.2.3: updated FCS_CKM.1(3) to comply with [FIPS186-2] 14. Section 6.2.4: updated FCS_CKM.1(4) to comply with [FIPS186-4] 15. Added new Section 6.2.6 "Cryptographic key destruction (COMM) (FCS_CKM.4_EXT)". All next Section will change their index. 16. Section 6.2.7 – updated SFR FCS_COP.1.2 for encryption, decryption, and integrity verification (LUKS) 17. Section 6.3.4 – modified the application note for FDP_UCT.1.1 18. Section 6.3.5 – modified the application note for FDP_UIT.1.1 19. Section 6.4.1 – modified the application note for FIA_AFL.1.2 20. Section 6.4.3 – modified the FIA_SOS.1.1 requirement to provide a better description for the provided authentication mechanisms 21. Section 6.4.5 – modified FIA_UAU.5.1 and FIA_UAU.5.2 to provide a better description for the provided authentication mechanisms 22. Section 6.5.15 – added application note to FMT_SAE.1 23. Section 6.5.17 – modified the application note for FMT_SMR.1 (changed wheel group with the admins group) 24. Section 6.6.2 – minor changes for the FPT_TST.1 25. Section 6.6.4 – minor changes for the FPT_FLS.1.1 26. Section 6.7.2 – modified the application note for FTA_TAB.1.1 27. Section 6.9.1 – updated table 8 28. Section 6.9.2 – updated table 9 29. Section 6.9.3 – updated table 10 30. Section 8.1.1 – provided a better description for the provided authentication mechanisms 31. Section 8.1.1.1 – provided a better description for dual-factor

			authentication 32. Section 8.1.1.2: • changed wheel group with the admins group • provided a better description for dual-factor authentication 33. Section 8.1.1.3 – provided a brief description for unlocking the session 34. Section 8.1.1.4 – provided a better description for dual-factor authentication 35. Section 8.1.6: • Added section 8.1.6.1 for “Secured network communication channels” • Added section 8.1.6.2 for “dded section 8.1.6.2 for tion channelsn 36. Section 8.1.7 – added restriction at boot time 37. Section 9.2: • removed reference to FIPS186-3 • added reference to FIP186-2 and FIPS186-4 38. Section 9.2: added reference RFC 6668

Common Criteria V3.1, rev. 4 – Security Target

Table of Contents

INDEX OF TABLES

INDEX OF FIGURES

1 Introduction

This document is a "Lite" version of the FINX RTOS SE v4.0 Security Target, PN 16200047415, 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 V4.0 Security Target**;
- Part number: **16200051033**;
- Revision: **03**;
- Date: **03/04/2017**.

1.2 TOE Identification

This ST applies to the CSCI identified as:

- Title: **FIN.X RTOS SE V4**;
- Part number: **16100031299**;
- Release: **01**;
- Abbreviation: **CSCI_FINXSE** or **TOE**.

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.

IPC Objects: Inter Process Communication (IPC) objects:

- Semaphores
- Shared memory

- Message queues
- Named pipes
- UNIX domain socket special files
- Signals.

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, including the security attributes defined by FIA_ATD.1;
- Audit records;
- 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 186-2 [**FIPS186-2**] RSA public/private key pairs and any associated passphrase;
- FIPS 186-4 [**FIPS186-4**] 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.

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 and creating and accessing files (e.g., output files of user-owned applications). The TOE uses discretionary access control as a mechanism to protect user data. Privileged commands are restricted to authorized administrators.

The TOE is capable of securely allocating resources to multiple users. These resources include multiple processors, memory, and attached peripheral and storage devices. The TOE facilitates controlled sharing of these resources based on subject and object security attributes. Many processes that are run on the TOE automatically, without requiring user interaction, do not have full privilege to the TOE itself. Although they are not bound to users, they are still subject to access control. The TOE is designed this way to enforce the principle of least privilege.

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.

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.

Discretionary access rights (e.g. read, write, execute) can be assigned to data objects with respect to subjects identified with their UID/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 TOE provides the following main security features:

- ✓ Identification and Authentication
- ✓ Auditing

- ✓ Discretionary Access Control
- ✓ Object-reuse functionality
- ✓ Security Management
- ✓ Cryptographic Services
- ✓ TSF Protection.

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 Intel Architecture platforms, with minimum requirements of:

- Microprocessor: Intel Pentium i686
- HD: SATA, 40GB
- RAM: 1.5GB
- network adapter or CD/DVD drive (only required for the TOE installation)
- USB port(s)
- Commercial Off the Shelf USB memory device

The hardware used during the evaluation process is:

- VP717/08x, VP917/08x, and VPB14/033-41E2-PTG: commercial, off-the-shelf and custom designed industrial computer boards for critical embedded applications. See the related data sheets at the URLs:
  - VPB14/033-41E2-PTG, http://www.gocct.com/sheets/VP/vpb1xmsd-rc.htm,
  - VP917/08x , http://www.gocct.com/sheets/VP/vp91xx1x.htm,
  - VP717/08x , http://www.gocct.com/sheets/VP/datasheet/vp71708x.pdf.
- VMWare ESXi v5.1.

All the 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.

TOE platform can be configured to support x86 32 and/or 64 bit applications. The evaluated configuration considers 32 bit and 64 bit applications.

1.6 TOE DESCRIPTION

1.6.1 Introduction

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 predictable execution context for supporting applications with real-time requirements. The TOE extends Linux native real-time support by integrating the Linux Kernel PREEMPT_RT patch **[PREEMPT_RT]**. The TOE provides such real-time enhancements in a security-oriented context.

This product evaluation covers a potentially distributed, but closed deployment 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 hardware, the BIOS firmware and potentially other firmware layers between the hardware and the TOE are considered to be part of the TOE environment.

A graphical user interface for TOE administration or any other operation is not included in the evaluated configuration.

Note that the TOE environment could include unprivileged applications that are not part of the evaluation. For example a network server using a port above 1024 may be used as a normal application running without root privileges on top of the TOE. The additional documentation specific for the evaluated configuration provides guidance how to set up such application on the TOE in a secure way.

1.6.2 Physical Scope

The TOE system software is the FIN.X RTOS SE V4 operating system as identified in section 1.2. The TOE and its documentation are supplied on ISO images and PDF files.

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 V4.0

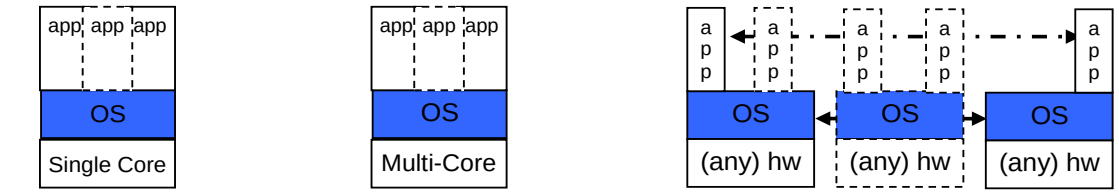
The hardware that is applicable to the evaluated configuration is listed in section 1.5.3. The analysis of the hardware capabilities as well as the firmware functionality is covered by this evaluation to the extent that the following capabilities supporting the security functionality are analysed and tested:

- Memory separation capability;
- Full testing of the security functionality on all listed hardware systems.

Physical peripheral devices (hard disks, network interface cards, CD/DVD drives, serial interfaces) can be used with the TOE without affecting its security functions.

Hot pluggable devices such as USB memory devices can be used for user's dual-factor authentication. When an USB memory device is required for dual-factor authentication, such a device shall be accessible by the TOE only at login time during authentication phase and then released. Other hot pluggable devices such as USB mice or keyboards can be used with the TOE by all the users without affecting its security functions, provided that they are connected before booting the operating system.

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 can 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.

1.6.3 Logical Scope

The primary security features of the TOE are described in the following paragraphs. The TOE provides many more 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 all forms of interactive login (e.g. using the SSH protocol or log in at the local console) as well as identity changes through the su or sudo command. These all rely on explicit authentication information provided interactively by a user.

- 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.
- The TOE authenticates the claimed identity of the user before allowing the user to perform any further actions.
- 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 TOE provides identification and authentication using the SSH V2 protocol [**SSH-4251**, **SSH-4252**, **SSH-4253**, **SSH-6668**] or using pluggable authentication modules (PAM) based upon user passwords or public/private key pairs (e.g., token based authentication). The quality of the passwords used can be enforced through configuration options. Other authentication methods (e.g., smart card authentication) are not part of the evaluated configuration.

The authentication security function allows the following authentication methods:

1. Password-based authentication – 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);
2. Dual-factor authentication uaIt can be used in conjunction with a successful password-based authentication during the log in process at the local console or when becoming another user (i.e. during 'su' actions);
3. Public-key-based authentication – It is used for initiating a SSH access without supplying the user's password.

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

1.6.3.2 Audit

The TOE provides an audit capability that allows generating audit records for security critical events. The authorized administrator can select which events are audited and for which users auditing is active. A list of events that can be audited is defined in section 6.1.

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 be taken when the threshold is exceeded. The possible actions include switching the operating system to single user mode (this prevents all user-initiated auditable actions) or halting the operating system.

The audit function also ensures that no audit records get lost due to exhaustion of the internal audit buffers. Processes that try to create an audit record while the internal audit buffers are full will be

halted until the required resources are available again. In the unlikely case of unrecoverable resource exhaustion, the kernel audit component initiates a kernel panic to prevent all further auditable events.

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 TOE includes the ext3 and ext4 file system, which supports POSIX ACLs. This allows defining access rights to files within this type of file system down to the granularity of a single user.

IPC objects use permission bits for discretionary access control.

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. The TOE supports secure wiping of hard disk devices (e.g., through the *shred* tool), but these tools are excluded from the evaluated configuration.

1.6.3.5 Security Management

The management of the security critical parameters of the TOE is performed by authorized administrators. A set of commands that require *root* privileges is used for TOE management. Security parameters are stored in specific files that are protected by the access control mechanisms of the TOE against unauthorized access by users that are not authorized administrators.

All authorized users are able to change their authentication data.

1.6.3.6 Cryptographic Services

The TOE provides cryptographically secured communication channels as well as cryptographic primitives [**FIPS140**, **FIPS186, OMSecPol**] that users (both privileged and unprivileged) can utilize for unspecified purposes. The TOE provides cryptographically secured communication to allow remote entities to log into the TOE; for interactive usage, the SSH V2 protocol is provided.

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.

1.6.3.8 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 V4.0 distribution, but do not contribute or interfere with the TSF.

File Systems: The following file system types are supported:

- the ext3 journaling filesystem,
- the ext4 journaling filesystem,
- the read-only ISO 9660 filesystem for CD-ROM drives and DVD drives,
- the process file system, procfs (/proc) represents processes / tasks as files and directories containing live status information for each process in the system. Process access decisions are enforced by DAC attributes inferred from the underlying process' DAC attributes. Additional restrictions apply for specific objects in this file system,
- the sysfs filesystem (sysfs) used to export and handle non-process related kernel information such as device driver specific information. Access to objects there can be restricted using the DAC mechanism (which consists of the permission bits only),
- the temporary filesystem (tmpfs) used as a fast non-persistent RAM based file system,
- the pseudo-terminal device file system (devpts) used to provide pseudo terminal support,
- the virtual root file system (rootfs) used temporarily during TOE start-up,
- the miscellaneous binary file format registration file system (binfmt_misc) used to configure interpreters for executing binary files based on file header information.

2 Conformance Claims

This ST is CC Part 2 extended and CC Part 3 conformant, with a claimed Evaluation Assurance Level of EAL4 augmented with ALC_FLR.1.

This Security Target does not claim conformance any Protection Profile, even if the **[OSPP]** and the **[GPOSPP]** have been used as a guideline.

Common Criteria **[CC]** for Information Technology Security Evaluation, Version 3.1 Revision 4, September 2012 has been taken as the basis for this conformance claim.

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.

3.1.3 Threats countered by the TOE

In the following the threats countered by the TOE:

T.AUDIT_COMPROMISE: A threat agent may view audit records, cause audit records to be lost or modified, or prevent future audit records from being recorded, thus masking a user's action.

T.MASQUERADE: A threat agent may masquerade as an authorized entity in order to gain unauthorized access to user data, TSF data, or TOE resources.

T.RESIDUAL_DATA: A threat agent may gain unauthorized access to user data or TSF data through reallocation of TOE resources from one user or process to another.

T.ASSETS_COMPROMISE: A threat agent may cause assets to be inappropriately accessed (viewed, modified or deleted).

T.UNAUTHORIZED_ACCESS: A threat agent may gain unauthorized access (view, modify, delete) to user data when the data is stored, processed, or transmitted.

T.TSFFUNC: A threat agent might use or modify functionality of the TSF without the necessary privilege to grant itself or others unauthorized access to TSF data or user data.

T.USER: A threat agent might gain access to user data, TSF data or TOE resources with the exception of public objects without being identified and authenticated.

T.TSFDATA: A threat agent might read or modify TSF data without the necessary authorization when the data is stored or transmitted.

T.USERDATA: A threat agent might gain access to user data at rest which is confidentiality protected without possessing the authorization of the owner, either at runtime of the TOE or when the TSF are inactive.

T.COMM: A threat agent might access a communication channel that establishes a trusted relationship between the TOE and a remote user or another remote trusted IT system or masquerade as another remote user or as another remote trusted IT system.

T.UNATTENDED_SESSION: A threat agent may gain unauthorized access to an unattended session.

T.UNIDENTIFIED_ACTIONS: The authorized administrator may fail to notice potential security violations. That could prevent the authorized administrator from taking action against a possible security violation.

3.2 ORGANIZATIONAL SECURITY POLICY

The organizational security policies addressed by the TOE are the following:

P.AUTHORIZED_USERS: Only those users who have been authorized to access the information within the TOE may access the TOE.

P.ACCESS_BANNER: The TOE shall display an initial banner describing restrictions of use, legal agreements, or any other appropriate information to which users must explicitly consent before accessing the TOE.

P.TRACE: The TOE shall provide the ability to review the actions of individual users.

P.EXPORT_ENCRYPT: The TOE shall provide the means to enforce encryption of exported information. Only AES and T-DES ciphers shall be provided by the TOE.

P.REMOTE_ENCRYPT: The TOE shall ensure only cryptographically-protected channels are employed for network connections.

P.I_AND_A: All users shall be identified and authenticated prior to accessing any controlled resources with the exception of public objects.

P.NEED_TO_KNOW: The organization must define a discretionary access control policy on a need-to-know basis which can be modelled based on:

- the owner of the object; and
- the identity of the subject attempting the access; and
- the implicit and explicit access rights to the object granted to the subject by the object owner or an authorized administrator.

Application Note: Being able to model an organization's access control policy based on the three properties above ensures that the organization's policy can be mapped to the security functions provided by the TOE. For example an access control policy based on time dependent or content dependent rules would not satisfy the above mentioned policy.

P.PWD_QUALITY: The TOE shall provide the means to enforce password quality. The TOE shall be configured to offer users:

- Password quality enforcement based on password string construction rules applied to passwords generated by the user,
- Passphrase quality enforcement based on passphrase string construction rules applied to passphrases generated by the user for LUKS-formatted devices.

P.PWD_PARAMS: The TOE shall provide the means to enforce password parameter limits as follows:

- minimum valid duration,
- maximum valid duration,
- maximum quantity of successive unsuccessful authentication attempts prior to account lockout,
- minimum password 'history depth',
- password validity expiration warning interval,
- nevertheless permit any user to change password at any time.

P.CRED_STORAGE: The TOE shall only store human user credentials in hash form, using SHA-512.

P.CRED_PROC: The TOE shall only authenticate human users by comparison of stored hash value with re-computed hash of user-entered credentials.

P.PERIPHERALS: The TOE shall provide the means to manage peripheral devices.

P.SYS_REL_1: The TOE shall provide the means to enforce accounting of security-related events, as a minimum:

- identification & authentication events (both Success & Failure and/or both Human and Machine users)
- all accounting log read accesses (S&F) (H&M)
- audit log [1] read access (S&F) (H&M)
- accounting log [2] storage capacity modification (S&F)
- audit log storage capacity modification (S&F)
- accounting log storage behaviour modification (S&F)
- audit log storage behaviour modification (S&F)
- modification to definition of events to be logged (S&F)
- modification to definition of events that cause alert to be emitted (S&F)
- filesystem mount operations (H&M) (S&F)
- modifications to data structures that represent machine and human users (S&F)
- modifications to data structures that represent access abilities over objects (S&F)
- human user adjustments to machine representation of time (S&F)
- changes to authentication parameters (S&F)
- changes to network configuration (S&F)
- uses of automated time interval behaviours [3] (S&F)

P.SYS_REL_2: The TOE shall provide the means to enforce accounting of security-related events, where such are generated by IT products in the environment:

- input validation failure
- message integrity failure
- message source confirmation failure[4]
- output validation failure
- uses of remote management behaviours

P.SYS_REL_3: When the TOE uses cryptographically-protected channels, it shall provide the means to enforce accounting of:

- remote access episodes (S&F)

1 - All the recorded occurred events.
2 - All the recorded occurred events related to the accounting operations like login, logout, *su* command, group changing etc. They are included in the audit events.
3 - It refers to any enabled time-based job scheduler, as *Cron*.
4 - Case of a mutual end point authentication between an IT product in the TOE environment and the TOE itself.

- identity of remote hosts
- any changes to security behaviour or configuration effected by the management episode.

3.3 SECURITY ASSUMPTIONS

The following assumptions are considered to be effective in TOE environment:

A.PHYSICAL: It is assumed that the IT environment provides the TOE with appropriate physical security, commensurate with the value of the IT assets protected by the TOE.

A.CONNECT: All connections to and from remote trusted IT systems and between physically-separate parts of the TSF not protected by the TSF itself are physically or logically protected within the TOE environment to ensure the integrity and confidentiality of the data transmitted and to ensure the authenticity of the communication end points.

A.MANAGE: The TOE security functionality is managed by one or more competent individuals. Those responsible for the TOE administration are not careless, wilfully negligent, or hostile, and will follow and abide by the instructions provided by the guidance documentation.

A.AUTHUSER: Authorized users possess the necessary authorization to access at least some of the information managed by the TOE and are expected to act in a cooperating manner in a benign environment.

A.TRAINEDUSER: Users are sufficiently trained and trusted to accomplish some task or group of tasks within a secure IT environment by exercising complete control over their user data.

A.DETECT: Any modification or corruption of security-enforcing or security-relevant files of the TOE, user data or the underlying platform caused either intentionally or accidentally will be detected by an authorized administrator.

A.PEER_MGT: All remote trusted IT systems trusted by the TSF to provide TSF data or services to the TOE, or to support the TSF in the enforcement of security policy decisions, are assumed to be under the same management control and operate under security policy constraints compatible with those of the TOE.

A.PEER_FUNC: All remote trusted IT systems trusted by the TSF to provide TSF data or services to the TOE, or to support the TSF in the enforcement of security policy decisions, are assumed to correctly implement the functionality used by the TSF consistent with the assumptions defined for this functionality.

4 Security Objectives

This section defines the security objectives for the TOE and its environment. These objectives are suitable to counter all identified threats and cover all identified organizational security policies and assumptions. The TOE security objectives are identified with "O." appended to the beginning of the name and the environment objectives are identified with "OE." appended to the beginning of the name.

4.1 TOE Security Objectives

O.ACCESS: the TOE must ensure that users gain only authorized access to it and to resources that it controls.

O.ACCESS_HISTORY: the TOE must display information (to authorized users) related to previous attempts to establish an interactive session or related to previous attempts to unlock a locked session.

O.AUDIT_GENERATION: the TOE must provide the capability to detect security relevant events and to create records of those events in the audit trail. The information recorded for security-relevant events must contain the time and date the event happened and, if possible, the identification of the user that caused the event, and must be in sufficient detail to help the authorized administrator detect attempted security violations or potential misconfiguration of the TOE security features that would leave the assets open to compromise.

O.AUDIT_PROTECTION: the TOE must provide the capability to protect audit information and to present audit information only to authorized administrators. The TOE must alert the authorized administrator of identified potential security violations.

O.AUDIT_REVIEW: the TOE must provide the capability to selectively view audit information.

O.CRYPTO_NET: the TOE must allow authorized users to remotely access the TOE using a cryptographically-protected network protocol that ensures integrity and confidentiality of the transported data and is able to authenticate the end points of the communication. Note that the same protocol may also be used in the case where the TSF is physically separated into multiple parts that must communicate securely with each other over untrusted network connections.

O.CRYPTO_MEDIA: the TOE must allow authorized administrator to securely export information using a cryptographically-protected protocol that ensures integrity and confidentiality of the exported data.

O.TRUSTED_CHANNEL: the TOE must be designed and implemented in a manner that allows for establishing a trusted channel between the TOE and a remote trusted IT system that protects the user data and TSF data transferred over this channel from disclosure and undetected modification and prevents masquerading of the remote trusted IT system.

O.DISCRETIONARY_ACCESS: the TOE must control access to resources based upon the identity of users and groups of users.

O.DISCRETIONARY_USER_CONTROL: the TOE must allow authorized users to specify which resources may be accessed by which users and groups of users.

O.DISPLAY_BANNER: during interactive log in, the TOE must display an advisory warning describing restrictions of use, legal agreements, or any other appropriate information to which users must explicitly consent before gaining shell's control.

O.ENFORCEMENT: the TOE must be designed and implemented in a manner which ensures that the organizational policies are enforced in the target environment. The TOE security policy is enforced in a manner which ensures that the organizational policies are enforced in the target environment, i.e., the integrity of the TSF is protected.

O.MANAGE: the TOE must provide all the functions and facilities necessary to support the authorized administrators in their management of the security of the TOE, and protect these functions and facilities from unauthorized use.

O.PROTECT: the TOE shall be able to protect the confidentiality of user data at rest separately for each user where the user can select the data which is being maintained under confidentiality protection.

O.RUNTIME.PROTECTION: the TOE shall offer a runtime protection mechanism for applications to mitigate the effects of buffer overruns potentially present in applications and associated libraries.

O.RESIDUAL_INFORMATION: the TOE must ensure that any data contained in a protected resource is not available when the resource is reallocated.

O.USER_AUTHENTICATION: the TOE must successfully verify the claimed identity of users before allowing any action the TOE has defined to provide to that user. After a successful password-based authentication method on a local terminal, the authorized user can be force to go through the dual-factor authentication before being able to open a new session.

O.USER_IDENTIFICATION: the TOE must uniquely identify users.

4.2 ENVIRONMENT SECURITY OBJECTIVES

The following objectives are to be met by the operational environment of the TOE.

OE.ADMIN: Those responsible for the TOE are competent and trustworthy individuals, capable of managing the TOE and the security of the information it contains.

OE.INFO_PROTECT: Those responsible for the TOE must establish and implement procedures to ensure that information is protected in an appropriate manner. In particular:

- All network and peripheral cabling must be approved for the transmittal of the most sensitive data held by the system. Such physical links are assumed to be adequately protected against threats to the confidentiality and integrity of the data transmitted,
- DAC protections on security-relevant files (such as audit trails and authentication databases) shall always be set up correctly,
- Users are authorized to access parts of the data managed by the TOE and are trained to exercise control over their own data.

OE.INSTALL: Those responsible for the TOE must establish and implement procedures to ensure that the hardware, software and firmware components that comprise the system are distributed, installed and configured in a secure manner supporting the security mechanisms provided by the TOE.

OE.MAINTENANCE: Authorized users of the TOE must ensure that the comprehensive diagnostics facilities provided by the product are invoked at every scheduled preventative maintenance period.

OE.PHYSICAL: Those responsible for the TOE must ensure that those parts of the TOE critical to enforcement of the security policy are protected from physical attack that might compromise IT security objectives. The protection must be commensurate with the value of the IT assets protected by the TOE.

OE.RECOVER: Those responsible for the TOE must ensure that procedures and/or mechanisms are provided to assure that after TOE failure or other discontinuity, recovery without a protection (i.e., security) compromise is achieved.

OE.TRUSTED.IT.SYSTEM: The remote trusted IT systems implement the protocols and mechanisms required by the TSF to support the enforcement of the security policy.
These remote trusted IT systems are under the same management domain as the TOE, are managed based on the same rules and policies applicable to the TOE, and are physically and logically protected equivalent to the TOE.

4.3 SECURITY OBJECTIVES RATIONALE

4.3.1 Security Objectives Coverage

Table 1 provides a mapping of TOE objectives to threats and policies, showing that each objective counters or enforces at least one threat or policy, respectively.

Objectives	Threats	OSPs
O.ACCESS	T.UNAUTHORIZED_ACCESS	P.NEED_TO_KNOW
O.ACCESS_HISTORY	T.UNAUTHORIZED_ACCESS	
O.AUDIT_GENERATION	T.AUDIT_COMPROMISE	P.SYS_REL_1, P.SYS_REL_2, P.SYS_REL_3
O.AUDIT_PROTECTION	T.AUDIT_COMPROMISE	
O.AUDIT_REVIEW	T.UNIDENTIFIED_ACTIONS	P.TRACE, P.SYS_REL_1, P.SYS_REL_2, P.SYS_REL_3
O.CRYPTO_NET	T.ASSETS_COMPROMISE, T.COMM	P.REMOTE_ENCRYPT
O.CRYPTO_MEDIA	T.ASSETS_COMPROMISE	P.EXPORT_ENCRYPT
O.TRUSTED_CHANNEL	T.COMM	
O.DISCRETIONARY_ACCESS	T.TSFFUNC, T.ASSETS_COMPROMISE, T.UNAUTHORIZED_ACCESS	P.NEED_TO_KNOW
O.DISCRETIONARY_USER_CONTROL	T.USER, T.ASSETS_COMPROMISE, T.UNAUTHORIZED_ACCESS	P.NEED_TO_KNOW
O.DISPLAY_BANNER		P.ACCESS_BANNER
O.ENFORCEMENT		P.AUTHORIZED_USERS, P.NEED_TO_KNOW
O.MANAGE	T.TSFFUNC, T.ASSETS_COMPROMISE	P.AUTHORIZED_USERS, P.NEED_TO_KNOW, P.PWD_PARAMS,

Objectives	Threats	OSPs
		P.PERIPHERALS, P.SYS_REL_1, P.SYS_REL_2, P.SYS_REL_3, P.TRACE
O.PROTECT	T.USERDATA, T.UNATTENDED_SESSION, T.UNAUTHORIZED_ACCESS	P.CRED_STORAGE P.NEED_TO_KNOW, P.PWD_QUALITY, P.PWD_PARAMS
O.RUNTIME.PROTECTION	T.TSFDATA T.USERDATA	
O.RESIDUAL_INFORMATION	T.RESIDUAL_DATA	
O.USER_AUTHENTICATION	T.USER, T.MASQUERADE	P.AUTHORIZED_USERS, P.CRED_PROC, P.I_AND_A, P.PWD_QUALITY, P.PWD_PARAMS
O.USER_IDENTIFICATION	T.USER, T.MASQUERADE	P.AUTHORIZED_USERS P.I_AND_A, P.SYS_REL_1, P.SYS_REL_2, P.SYS_REL_3

Table 1: Mapping of security objectives to threats and policies

Table 2 provides a mapping of the objectives for the Operational Environment to assumptions, threats and policies, showing that each objective holds, counters or enforces at least one assumption, threat or policy, respectively.

Objectives	Assumptions	Threats	OSPs
OE.ADMIN	A.AUTHUSER, A.MANAGE, A.TRAINEDUSER		
OE.INFO_PROTECT	A.AUTHUSER, A.CONNECT,		

Objectives	Assumptions	Threats	OSPs
	A.MANAGE, A.PHYSICAL, A.TRAINEDUSER		
OE.INSTALL	A.DETECT A.MANAGE		
OE.MAINTENANCE	A.DETECT		
OE.PHYSICAL	A.PHYSICAL	T.AUDIT_COMPROMISE, T.ASSETS_COMPROMISE, T.UNAUTHORIZED_ACCESS	
OE.RECOVER	A.DETECT A.MANAGE		
OE.TRUSTED.IT.SYSTEM	A.CONNECT, A.PEER_FUNC, A.PEER_MGT	T.COMM	

Table 2: Mapping of security objectives for the Operational Environment to assumptions, threats and policies

4.3.2 Security Objectives Sufficiency

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

Threats	Rationale for security objectives
T.AUDIT_COMPROMISE	O.AUDIT_GENERATION provides the capability to detect and create records of security relevant events. Audit records identify the user responsible for the event and are an important form of evidence that can be used to track an attacker's actions. Tampering with or destruction of audit data by physical means is addressed by OE.PHYSICAL, which provides physical security controls to the TOE environment. O.AUDIT_PROTECTION provides the capability to specifically protect audit information from external interference, tampering, or unauthorized disclosure.
T.MASQUERADE	To address this threat, O.USER_IDENTIFICATION uniquely identifies the user as a legitimate user and O.USER_AUTHENTICATION authenticates this user preventing unauthorized users, processes, or external IT entities from masquerading as an authorized entity.
T.RESIDUAL_DATA	The sharing of hardware resources such as primary and secondary

Threats	Rationale for security objectives
	storage components between users introduces the potential for information flow in violation of the TOE security policy when hardware resources are de-allocated from one user and allocated to another. In order to prevent such unintended consequences, the TOE counters the compromise of the TOE security policy through mechanisms that ensure that residual information cannot be accessed after the resource has been reallocated (O.RESIDUAL_INFORMATION). The intent here is to prevent the unauthorized flow of information that would violate the TOE security policy. The intent is not to require explicit scrubbing or overwriting of data prior to reuse of the storage resource. Therefore, the presence of "residual" data in a storage resource is acceptable as long as it cannot be accessed by subsequent users such that a violation of the TOE security policy results.
T.ASSETS_COMPROMISE	The tampering with or destruction of TSF hardware, software, or configuration data via physical means is addressed by the physical security controls present in the TOE environment (OE.PHYSICAL). • O.MANAGE mitigates this threat providing the authorized users with the facilities necessary to manage TOE security. • O.DISCRETIONARY_ACCESS avoids a resource to be inappropriately accessed by unauthorized user. • O.DISCRETIONARY_USER_CONTROL mitigates this threat providing the authorized users with the capability to specify by which user an asset may be accessed. • O.CRYPTO_NET mitigates this threat by requiring cryptographically-protected communication channels for data including TSF data controlled by the TOE in transit between trusted IT systems. • O.CRYPTO_MEDIA mitigates this threat by requiring cryptographically-protected export of information, including TSF data controlled by the TOE.
T.COMM	The threat of accessing a communication channel that establishes a trusted relationship between the TOE and another remote user or trusted IT system is addressed by: • O.TRUSTED_CHANNEL requiring that the TOE implements a trusted channel between itself and a remote trusted IT system protecting the user data and TSF data transferred over this channel from disclosure and undetected modification, and prevents masquerading of the remote trusted IT system, • O.CRYPTO_NET requiring the users remote access protected by

Threats	Rationale for security objectives
	a cryptographic network protocol. • OE.TRUSTED.IT.SYSTEM requiring that those systems providing the functions required by the TOE are trusted and sufficiently protected from any attack that may cause those functions to provide bad results.
T.UNATTENDED_SESSION	When an authorized user leaves an active session unattended, an unauthorized user may gain access to the unattended session. • O.PROTECT mitigates this threat by providing mechanisms to protect user data and resources from unauthorized access by ensuring that the TSF will lock an interactive session and make the visible contents unreadable after a specified time interval of session inactivity.
T.UNAUTHORIZED_ACCESS	Unauthorized users may physically access TOE resources. To mitigate this threat, OE.PHYSICAL restricts the physical access only to authorized personnel. Within the computing environment, O.ACCESS restricts all access controls to authorized users based on their user identity. At the same time, O.PROTECT enforces access rules by providing mechanisms to protect the user data from unauthorized disclosure and modification. Moreover: • O.ACCESS_HISTORY helps authorized users confirming- their previously established session or may help detecting possible unsuccessful attempts to their account by an unauthorized user. • O.DISCRETIONARY_ACCESS and O.DISCRETIONARY_USER_CONTROL allow the access control for resources based upon the identities of the users and their groups and the ability for authorized users to specify the access to their resources. This ensures that a user can access a resource only if the requested type of access has been granted by the user responsible for the management of access rights to the resource.
T.UNIDENTIFIED_ACTIONS	The threat of an authorized administrator failing to notice security violation on audit events may occur. To mitigate this threat, O.AUDIT_REVIEW provides the capability to selectively view audit information, and alert the authorized administrator of identified potential security violations.
T.TSFFUNC	The threat of accessing TSF functions without proper authorization is removed by: • O.CRYPTO_NET requiring cryptographically-protected communication channels to limit which TSF functions are

Threats	Rationale for security objectives
	accessible to external entities. • O.MANAGE requiring that only authorized users utilize management TSF functions.
T.USER	The threat of accessing user data, TSF data or TOE resources without being identified and authenticated is removed by: • O.USER_IDENTIFICATION and O.USER_AUTHENTICATION requiring that each entity interacting with the TOE is properly identified and authenticated before allowing any action the TOE has defined to provide to authenticated users only.
T.USERDATA	The threat of accessing user data without proper authorization is removed by: • O.CRYPTO_NET requiring cryptographically-protected communication channels for data including user data controlled by the TOE in transit between trusted IT systems. • O.DISCRETIONARY_ACCESS requiring that data including user data stored with the TOE, have discretionary access control protection. • O.DISCRETIONARY_USER_CONTROL requiring the TSF to mediate communication between subjects. • O.RUNTIME.PROTECTION requiring the TSF to provide functionality to mitigate the effect of protentially present buffer overrun dedicated TSF applications and their libraries.
T.TSFDATA	The threat of accessing TSF data without proper authorization is removed by: • O.CRYPTO_NET requiring cryptographically-protected communication channels for data including TSF data controlled by the TOE in transit between trusted IT systems. • O.DISCRETIONARY_ACCESS requiring that data, including TSF data stored with the TOE, have discretionary access control protection. • O.RUNTIME.PROTECTION requiring the TSF to provide functionality to mitigate the effect of protentially present buffer overrun dedicated TSF applications and their libraries.

Table 3: Sufficiency of objectives countering threats

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

objectives for the environment that trace back to an assumption are achieved, the intended usage is supported:

Assumptions	Rationale for security objectives
A.PHYSICAL	The assumption on the IT environment to provide the TOE with appropriate physical security, commensurate with the value of the IT assets protected by the TOE is covered by: • OE.INFO_PROTECT: requiring the approval of network and peripheral cabling, • OE.PHYSICAL: requiring physical protection.
A.CONNECT	The assumption that all connections to and from remote trusted IT systems and between physically separate parts of the TSF not protected by the TSF itself are physically or logically protected is covered by: • OE.TRUSTED.IT.SYSTEM: demanding the physical and logical protection equivalent to the TOE, • OE.INFO_PROTECT: requiring security procedures for network physical links, DAC policy and users training.
A.MANAGE	The assumptions on the TOE security functionality being managed by one or more trustworthy individuals is covered by: • OE.ADMIN: ensuring that those responsible for the TOE are competent and trustworthy individuals, capable of managing the TOE and the security of the information it contains, • OE.INFO_PROTECT: requiring that DAC protections on security-relevant files (such as audit trails and authentication databases) shall always be set up correctly and that users are authorized to access parts of the data maintained by the TOE., • OE.INSTALL: requiring personnel to ensure that components that comprise the system are distributed, installed and configured in a secure manner supporting the security mechanisms provided by the TOE, • OE.RECOVER: requiring personnel to perform all the required actions to bring the TOE into a secure state after a TOE failure or discontinuity.
A.AUTHUSER	The assumption on authorized users to possess the necessary authorization to access at least some of the information managed by the TOE and to act in a cooperating manner in a benign environment is covered by: • OE.ADMIN: ensuring that those responsible for the TOE are competent and trustworthy individuals, capable of managing

Assumptions	Rationale for security objectives
	the TOE and the security of the information it contains, • OE.INFO_PROTECT: requiring that DAC protections on security-relevant files (such as audit trails and authentication databases) shall always be set up correctly and that users are authorized to access parts of the data maintained by the TOE.
A.TRAINEDUSER	The assumptions on users to be sufficiently trained and trusted to accomplish some task or group of tasks within a secure IT environment by exercising complete control over their user data is covered by: • OE.ADMIN: ensuring that those responsible for the TOE are competent and trustworthy individuals, capable of managing the TOE and the security of the information it contains, • OE.INFO_PROTECT: requiring that those responsible for the TOE must establish and implement procedures to ensure that information is protected in an appropriate manner and that users are trained to exercise control over their own data.
A.DETECT	The assumption that modification or corruption of security-enforcing or security-relevant files will be detected by an authorized administrator is covered by: • OE.INSTALL: requiring personnel to ensure that components that comprise the system are distributed, installed and configured in a secure manner supporting the security mechanisms provided by the TOE, • OE.MAINTENANCE: requiring an authorized users to ensure that the diagnostics facilities are invoked at every scheduled preventative maintenance period, verifying the correct operation of the TOE, • OE.RECOVER: requiring personnel to perform all the required actions to bring the TOE into a secure state after a TOE failure or discontinuity..
A.PEER_MGT	The assumption on all remote trusted IT systems to be under the same management control and operate under security policy constraints compatible with those of the TOE is covered by: • OE.TRUSTED.IT.SYSTEM: requiring that these remote trusted IT systems are under the same management domain as the TOE, and are managed based on the same rules and policies applicable to the TOE.
A.PEER_FUNC	The assumption on all remote trusted IT systems to correctly implement the functionality used by the TSF consistent with the

Assumptions	Rationale for security objectives
	assumptions defined for this functionality is covered by: • OE.TRUSTED.IT.SYSTEM: requiring that the remote trusted IT systems implement the protocols and mechanisms required by the TSF to support the enforcement of the security policy.

Table 4: Sufficiency of objectives holding Assumptions

The following rationale provides justification that the security objectives are suitable to cover each individual organizational security policy, that each security objective that traces back to an OSP, when achieved, actually contributes to the implementation of the OSP, and that if all security objectives that trace back to an OSP are achieved, the OSP is implemented:

OSP	Rationale for security objectives
P.AUTHORIZED_USERS	The policy demanding that users have to be authorized for access to the TOE is implemented by: • O.USER_IDENTIFICATION&O.USER_AUTHENTICATION: allowing that only identified and authenticated users gain access to the TOE and its resources, • O.MANAGE: allowing the management of this functions, • O.ENFORCEMENT: ensuring the correct invocation of the functions.
P.ACCESS_BANNER	The policy to ensure an initial banner with some access restrictive information is displayed is implemented by: • O.DISPLAY_BANNER: ensuring that the TOE displays a banner that provides authorized users with an advisory warning about the unauthorized use of the TOE.
P.PERIPHERALS	The policy to ensure the means to manage peripheral devices is implemented by: • O.MANAGE: allowing the management of physically connected devices.
P.SYS_REL_1	The policy to ensure accounting of the security-relevant events within the TOE is implemented by: • O.AUDIT_GENERATION: providing the capability to detect security relevant events and create records of those events in the audit trail, • O.AUDIT_REVIEW: providing the capability to selectively review potential security violations, • O.MANAGE: allowing the management of security attributes, • O.USER_IDENTIFICATION: providing a unique users

OSP	Rationale for security objectives
	identification.
P.SYS_REL_2	The policy to ensure accounting of the security-relevant events, where such are generated by IT products in the environment, is implemented by: • O.AUDIT_GENERATION: providing the capability to detect security relevant events and create records of those events in the audit trail, • O.AUDIT_REVIEW: providing the capability to selectively review potential security violations, • O.MANAGE: allowing the management of security attributes, • O.USER_IDENTIFICATION: providing a unique users identification.
P.SYS_REL_3	The policy to ensure accounting of the security-relevant events when the TOE is configured with cryptographic remote management channels is implemented by: • O.AUDIT_GENERATION: providing the capability to detect security relevant events and create records of those events in the audit trail, • O.AUDIT_REVIEW: providing the capability to selectively review potential security violations, • O.MANAGE: allowing the management of security attributes, • O.USER_IDENTIFICATION: providing a unique users identification.
P.EXPORT_ENCRYPT	The policy to ensure the encryption of exported information is implemented by: • O.CRYPTO_MEDIA: enforcing the use of a cryptographically-protected protocol that ensures integrity and confidentiality of the exported data.
P.REMOTE_ENCRYPT	The policy to ensure the encryption of remote management channels is implemented by: • O.CRYPTO_NET: enforcing the use of a cryptographically-protected network protocol that ensures integrity and confidentiality of the transported data and the authentication of the end points of the communication.
P.I_AND_A	The policy to ensure all users must be identified and authenticated prior to accessing any controlled resources with the exception of public objects is implemented by: • O.USER_AUTHENTICATION: requiring the TOE verify the claimed

OSP	Rationale for security objectives
	identity of users, • O.USER_IDENTIFICATION: requiring the TOE uniquely identify users.
P.NEED_TO_KNOW	The policy to restrict access to and modification of information to authorized users which have a "*need to know*" for that information is implemented by: • O.ACCESS: ensuring that users gain only authorized access to the TOE and to resources that it controls, • O.DISCRETIONARY_ACCESS: controlling the resources based on the identity of users, • O.DISCRETIONARY_USER_CONTROL: allowing authorized users (the objects owners) to specify the named objects may be accessed by which users and groups of users, • O.PROTECT: providing mechanisms to protect user data and resources, • O.MANAGE: allowing authorized users to manage these functions, • O.ENFORCEMENT: ensuring that the functions are invoked and operate correctly.
P.TRACE	The policy to provide the ability to review the actions of individual users is implemented by: • O.AUDIT_REVIEW: providing the capability to accurately and selectively review audit information at the individual user level and alert the authorized administrator of identified potential security violations. • O.MANAGE: allowing users to manage these functions.

P.PWD_QUALITY	The policy to provide the means to enforce password quality is implemented by:
	• O.PROTECT: enforcing access rules by providing mechanisms to prevent the user to provide password with low quality that may lead to unauthorized data disclosure and modification
	• O.USER_AUTHENTICATION:
	  ○ ensuring authentication of users through the password mechanism,
	  ○ ensuring secure access to LUKS-formatted devices through the passphrase mechanism.

P.PWD_PARAMS	The policy to provide the means to enforce password parameter limits is implemented by:

OSP	Rationale for security objectives

	• O.PROTECT: enforcing users to change password, thus reducing the risk of unauthorized data disclosure and modification,
	• O.USER_AUTHENTICATION:
	  ○ ensuring authentication of users through the password mechanism,
	  ○ ensuring secure access to LUKS-formatted devices through the passphrase mechanism.
	• O.MANAGE: allowing the authorized administrator to set all the parameters specified by the policies.

P.CRED_STORAGE	The policy to enforce the storage of human user credentials in hash form is implemented by: • O.PROTECT: considering user credentials as user assets, this objective provides mechanisms to protect this assets.
P.CRED_PROC	The policy to enforce the comparison of stored hash value with re-computed hash of user-entered credentials is implemented by: • O.USER_AUTHENTICATION: ensuring the verification of the claimed users credential.

Table 5: Sufficiency of objectives enforcing Organizational Security Policies

5 Extended Functional Requirements

Table 6 summarizes the extended functional requirements defined for the TOE. These additional functional requirements, with short name ending in "**_EXT**", are detailed in next subsections.

Extended Component	Component behaviour name
FCS_CKM.4_EXT	Cryptographic key destruction

Table 6: Extended functional Requirements

5.1 CLASS FCS: CRYPTOGRAPHIC SUPPORT

5.1.1 Cryptographic key destruction (FCS_CKM.4_EXT)

This extended component respects the FCS_CKM.4 component to which it is inspired. The extension was required because cryptographic key destruction requires cryptographic keys to be destroyed when no longer required.

5.1.1.1 FCS_CKM.4_EXT – Cryptographic key destruction

Hierarchical to: No other components.

Dependencies: [FDP_ITC.1 Import of user data without security attributes, or FDP_ITC.2 Import of user data with security attributes, or FCS_CKM.1 Cryptographic key generation].

FCS_CKM.4.1_EXT The TSF shall zeroize cryptographic keys when no longer required.

6 Security Functional Requirements

This section contains detailed security functional requirements for the TOE.

All operations are marked in **bold** within each of the requirements regardless if they have already been defined as instantiations in one of the Protection Profiles or not. When **Refinements** made by ST author imply substitutions the deleted text is highlighted with strikethrough expression.

6.1 SECURITY AUDIT (FAU)

6.1.1 Audit Data Generation (FAU_GEN.1)

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

- Start-up and shutdown of the audit functions,
- all auditable events for the **basic** level of audit**, listed in column "Events" of Table 7 (Auditable Events), except FIA_UID.2's user identity during failures.**

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

- Date and time of the event, type of event, subject identity (if applicable) and the outcome (success or failure) of the event,
- For each audit event type, based on the auditable event definitions of the functional components included in the ST:
  - **host identification (if applicable),**
  - **changes in system operational mode (if applicable)**

Security Functional Requirements	Events
Security Audit (FAU)	
Audit Review (FAU_SAR.1)	Reading of information from the audit records (name of the object).
Restricted Audit Review (FAU_SAR.2)	Unsuccessful attempts to read information from the audit records.
Selectable Audit Review (FAU_SAR.3)	None.
Selective Audit (FAU_SEL.1)	All modifications to the audit configuration that occur while the audit collection functions are operating.

Security Functional Requirements	Events
Protected Audit Trail Storage (FAU_STG.1)	None.
Action in case of possible audit data loss (FAU_STG.3)	Actions taken due to exceeding of a threshold (message sent to the authorized administrator).
Prevention of audit data loss (FAU_STG.4)	Actions taken due to the audit storage failure (message sent to the authorized administrator).
Cryptographic Support (FCS)	
Cryptographic key generation (Symmetric Keys) (FCS_CKM.1(1))	None.
Cryptographic key generation (LUKS) (FCS_CKM.1(2))	None.
Cryptographic key generation (RSA) (FCS_CKM.1(3))	None.
Cryptographic key generation (DSA) (FCS_CKM.1(4))	None.
Cryptographic key distribution (COMM) (FCS_CKM.2)	None.
Cryptographic key destruction (FCS_CKM.4.1_EXT)	None.
Cryptographic operation (COMM) (FCS_COP.1(1))	None.
Cryptographic Operation (LUKS) (FCS_COP.1(2))	None.
User Data Protection (FDP)	
Subset Access Control Policy (FDP_ACC.1)	None.
Security Attribute Based Access Control (File System Objects) (FDP_ACF.1(1))	All requests to perform an operation on an object(File System Objects) covered by the SFP (the name of the object being accessed).
Security Attribute Based Access Control	All requests to perform an operation on an object (IPC Objects) covered by the SFP (the name of the object

Security Functional Requirements	Events
(IPC Objects) (FDP_ACF.1(2))	being accessed).
Full Residual Information Protection (FDP_RIP.2)	None.
Basic data exchange confidentiality (FDP_UCT.1)	None.
Data exchange integrity (FDP_UIT.1)	None.
Identification and Authentication (FIA)	
Authentication Failure Handling (FIA_AFL.1)	The reaching of the threshold for the unsuccessful authentication attempts. The action taken (disable for authorized non-administrative users, delay for authorized administrators). The re-enablement of disabled non-administrative accounts.
User Attribute Definition (FIA_ATD.1)	None.
Verification of Secrets (FIA_SOS.1)	Rejection or acceptance by the TSF of any tested secret.
Authentication (FIA_UAU.2)	All uses of the authentication mechanism.
Multiple authentication mechanisms (FIA_UAU.5)	All uses of the authentication mechanism.
Re-authenticating (FIA_UAU.6)	All re-authentication mechanism attempts when changing authentication data (e.g., terminal identifier, source IP address).
Protected Authentication Feedback (FIA_UAU.7)	None.
Identification (FIA_UID.2)	All uses of the user identification mechanism, including the entity provided during successful attempts.
User-Subject Binding (FIA_USB.1)	Success and failure of binding user security attributes to a subject (e.g. success and failure to create a subject).

Security Functional Requirements	Events
Security Management (FMT)	
Management of Security Functions Behavior (for specification of auditable events) (FMT_MOF.1(1))	All modifications in the behaviour of the functions in the TSF.
Management of Security Functions Behavior (for authentication data) (FMT_MOF.1(2))	All modifications in the behaviour of the functions in the TSF.
Management of Object Security Attributes (FMT_MSA.1)	All modifications of the values of security attributes.
Secure Security Attributes (FMT_MSA.2)	All offered and rejected values for a security attribute.
Static Attributes Initialization (FMT_MSA.3)	Modifications of the default setting of permissive or restrictive rules. All modifications of the initial value of security attributes.
Management of TSF Data (for general TSF data) (FMT_MTD.1 – all iterations)	All modifications to the values of TSF data. (where applicable)
Revocation (to authorized administrators) (FMT_REV.1(1))	All revocation of security attributes (the security attributes that have been revoked).
Revocation (to owners and authorized administrators) (FMT_REV.1(2))	All revocation of security attributes (the security attributes that have been revoked, the object with which the security attributes are associated).
Time-limited authorization (FMT_SAE.1)	Specification of the expiration time for an attribute. Action taken due to attribute expiration.
Specification of Management Functions (FMT_SMF.1)	None (covered by other management functions).
Security Roles (FMT_SMR.1)	Modifications to the group of users that are part of a role (the role the user is associated with or disassociated from).
Protection of the TOE Security Functions (FPT)	
Reliable Time Stamps (FPT_STM.1)	Changes to the time.

Security Functional Requirements	Events
TSF Self Test (FPT_TST.1)	Execution of the TSF self tests and the results of the tests.
Failure with preservation of secure state - full buffer overflow protection (FPT_FLS.1(1))	None.
Failure with preservation of secure state - partial buffer overflow protection (FPT_FLS.1(2))	None.
TOE Access (FTA)	
User-Initiated Locking (FTA_SSL.2)	Any attempts at unlocking an interactive session.
Default TOE Access Banners (FTA_TAB.1)	None.
TOE Access History (FTA_TAH.1)	None.
Trusted path/channels (FTP)	
Inter-TSF trusted channel (FTP_ITC.1)	Set-up of trusted channel.

Table 7: Auditable events

6.1.2 User Identity Association (FAU_GEN.2)

FAU_GEN.2.1 For audit events resulting from actions of identified users, the TSF shall be able to associate each auditable event with the identity of the user that caused the event.

Application Note: *The TOE maintains a "Login UID", which is inherited by every new process spawned. This allows the TOE to identify the "real" originator of an event, regardless if he has changed his real and / or effective and filesystem UID e. g. using the su command or executing a setuid or setgid program.*

6.1.3 Audit Review (FAU_SAR.1)

FAU_SAR.1.1 The TSF shall provide **authorized administrators** with the capability to read **all audit information** from the audit records.

FAU_SAR.1.2 The TSF shall provide the audit records in a manner suitable for the user to interpret the information.

6.1.4 Restricted Audit Review (FAU_SAR.2)

FAU_SAR.2.1 The TSF shall prohibit all users read access to the audit records, except those users that have been granted explicit read-access.

Application Note: *DAC permissions ensure that only authorized administrators have access to the audit records.*

6.1.5 Selectable Audit Review (FAU_SAR.3)

FAU_SAR.3.1 The TSF shall provide the ability to apply **searches, sorting and ordering** of audit data based on **the following attributes**:

- Date and time of the event;
- Type of event;
- Event ID;
- User identity.

6.1.6 Selective Audit (FAU_SEL.1)

FAU_SEL.1.1 **Refinement:** The TSF shall be able to ~~select the set of events to be audited~~ **include or exclude auditable events** from the set of ~~all auditable~~ **audited** events based on the following attributes:

- **Type of audit event;**
- **Subject (process ID) or user identity;**
- **Outcome (success or failure) of the audit event;**
- **Object identity;**
- **Access types on a particular object;**
- **System call number.**

Application Note: *to make the requirement clearer, the words "select the set of audited" were replaced with "include or exclude auditable" and the word "auditable" was replaced with "audited".*

Application Note: *The TOE provides an application that allows specification of the audit rules which injects the rules into the kernel for enforcement. The Linux kernel auditing mechanism obtains all audit events and decides based on this rule set whether an event is forwarded to the audit daemon for storage.*

6.1.7 Protected Audit Trail Storage (FAU_STG.1)

FAU_STG.1.1 The TSF shall protect the stored audit records in the audit trail from unauthorized deletion.

FAU_STG.1.2 **Refinement:** The TSF shall be able to prevent unauthorized modifications to the stored audit records in the audit trail.

Application note: *DAC permissions ensure that audit trail is readable and writeable to the authorized administrators only. The word "unauthorized" was then deleted, since no one can be authorized to modify audit records.*

6.1.8 Action in case of possible audit data loss (FAU_STG.3)

FAU_STG.3.1 The TSF shall **notify an authorized administrator of the possible audit data loss** if the audit trail exceeds **an authorized administrator selectable, pre-defined size limit**.

Application note: *The alarm generated by the TOE can be configured to be a syslog message or the execution of an administrator-specified application. This message or action of executing the application is generated when the audit trail capacity exceeds the limit defined in the auditd.conf file*.

6.1.9 Prevention of Audit Data Loss (FAU_STG.4)

FAU_STG.4.1 **Refinement**: The TSF shall **be able to prevent audited events, except those taken by the authorised ~~user with special rights~~ administrator**, and **no other action** if the audit trail is full.

Application Note: *If the audit trail stored on disk gets full, the audit daemon will execute an audit administrator defined action. The possible actions include a switch to single user mode or operating system halt, each of these will terminate all processes capable of generating auditable events. The authorized administrator can then back up the audit trail and make space available for the audit trail, then restart the TOE in multiuser mode. If TOE is used in mission-critical contexts, the described behaviour can be disabled by an authorized administrator.*

6.2 CRYPTOGRAPHIC SUPPORT (FCS)

6.2.1 Cryptographic key generation (Symmetric Keys) (FCS_CKM.1(1))

FCS_CKM.1(1).1 The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm **as specified in product specific** and specified cryptographic key sizes:

- **AES: 128 bits, 192 bits, and 256 bits**
- **T-DES: 168 bits**

that meet the following: **not specified.**

Application Note: *integrity protection to generated keys is provided in the following manner:*

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

Application Notes: *Only the required key size is specified. The OpenSSH applications (when ssh client is invoked, the ssh-keygen application is used, or a new SSH connection is processed by sshd) use the PRNG from the OpenSSL library to generate random numbers. The TOE provides /dev/urandom and /dev/random random entropy pool devices. The OpenSSL library makes sure that the PRNG state is unique for each thread by seeding the PRNG via /dev/urandom as first choice and, if such source of unpredictable data is not available, it will also try the /dev/random.*

6.2.2 Cryptographic key generation (LUKS)(FCS_CKM.1(2))

FCS_CKM.1(2).1 **Refinement:** The TSF shall generate symmetric cryptographic keys in accordance with a specified cryptographic key generation algorithm using PBKDF2 as defined in RFC 2898 [**PBKDF2**], and specified cryptographic key sizes:

- **128 bits,**
- **256 bits.**

that meet the following: RFC 2898 [PBKDF2].

6.2.3 Cryptographic key generation (RSA)(FCS_CKM.1(3))

FCS_CKM.1(3).1 **Refinement:** The TSF shall generate RSA cryptographic keys in accordance with a specified cryptographic key generation algorithm defined in U.S. NIST FIPS PUB 186-2 [**FIPS186-2**] appendix B.3 and specified cryptographic key sizes:

- **1024 bits**
- **1536 bits**
- **2048 bits**
- **3072 bits**
- **4096 bits**

that meet the following:

- **U.S. NIST FIPS PUB 186-2**

Application Note: *The TOE supports the generation of RSA keys for the OpenSSH user keys using the OpenSSL application or the ssh-keygen(1) application.*

RSA keys for dual-factor authentication are generated by the OpenSSL application.

6.2.4 Cryptographic key generation(DSA) (FCS_CKM.1(4))

FCS_CKM.1(4).1 **Refinement:** The TSF shall generate **DSA** cryptographic keys in accordance with a specified cryptographic key generation algorithm **defined in U.S. NIST FIPS PUB 186-4[FIPS186-4] appendix B.1** and specified cryptographic key sizes:

- **L=1024 bits, N=160 bits;**
- **L=2048 bits, N=224 bits;**
- **L=2048 bits, N=256 bits;**
- **L=3072 bits, N=256 bits;**

that meet the following:

- **U.S. NIST FIPS PUB 186-4**

Application Note: *The TOE supports the generation of DSA keys for the OpenSSH user keys using the OpenSSL application or the ssh-keygen(1) application.*

DSA keys for dual-factor authentication are generated by the OpenSSL application.

6.2.5 Cryptographic key distribution (COMM) (FCS_CKM.2)

FCS_CKM.2.1 The TSF shall distribute cryptographic keys in accordance with a specified cryptographic key distribution method:

- **Diffie-Hellman key agreement method defined for the SSH protocol by RFC4253;**
- **Public DSS, RSA host key exchange defined for the SSH protocol by RFC4253.**

Application Note: *DSS defined in RFC4253 [**SSH-4253**] for the host key exchange is compliant with DSA defined in FIPS 186-4.*

6.2.6 Cryptographic key destruction (COMM) (FCS_CKM.4_EXT)

FCS_CKM.4.1_EXT The TSF shall zeroize cryptographic keys when no longer required.

Application Note: *Keys residing in internally allocated data structures can only be accessed using the cryptographic module defined API. The operating system protects memory and process space from unauthorized access. Zeroization of sensitive data is performed automatically by API function calls for intermediate data items, and on demand by the calling process using cryptographic module provided API function calls provided for that purpose. Only the process that creates or imports keys can use or export them. No persistent storage of key data is performed by the cryptographic module. All API functions are executed by the invoking process in a non-overlapping sequence such that no two API functions will execute concurrently.*

The calling process can perform key zeroization of keys by calling an API function.

Application Note: *API functions are provided by OpenSSL library for zeroizing symmetric keys and by Cryptsetup library for zeroizing LUKS keys.*

6.2.7 Cryptographic operation (COMM) (FCS_COP.1(1))

FCS_COP.1(1).1 **Refinement:** The TSF shall perform **encryption, decryption, and integrity verification,** in accordance with ~~a specified~~ **the following** cryptographic algorithms**,** ~~and~~ cryptographic key sizes**,** ~~that meet the following~~ **and applicable standards:**

- **SSH allowing the use of AES in CBC mode with 128 bits, 192 bits and 256 bits key size, and HMAC-SHA1, HMAC-SHA2-256, and HMAC-SHA2-512, defined by RFC 4253 and RFC 6668;**
- **SSH allowing the use of AES in CTR mode with 128 bits, 192 bits and 256 bits key size, and HMAC-SHA1, HMAC-SHA2-256, and HMAC-SHA2-512, defined by RFC 4253 and RFC 6668**
- **SSH allowing the use of TDES in CBC mode with 168 bits key size, and HMAC-SHA1, HMAC-SHA2-256, and HMAC-SHA2-512, defined by RFC 4253 and RFC 6668;**
- **SSH allowing the use of DSA with:**
  - **L=1024 bits, N=160 bits;**
  - **L=2048 bits, N=224 bits;**
  - **L=2048 bits, N=256 bits;**
  - **L=3072 bits, N=256 bits.**

  with the format of ssh-dss for "public-key" authentication defined by RFC 4252;

- **SSH allowing the use of RSA with key sizes of 1024 bits, 1536 bits, 2048 bits, 3072 bits, and 4096 bits, with the format of ssh-rsa for "public-key" authentication defined by RFC 4252.**

6.2.8 Cryptographic Operation (LUKS) (FCS_COP.1(2))

FCS_COP.1(2).1 **Refinement:** The TSF shall perform **encryption, decryption, and integrity verification** in accordance with a specified cryptographic algorithm formed with any permutation of the following types of cryptographic primitives [LUKS]:

- Ciphers: **AES, T-DES**
- Message Digest: **SHA-1, SHA-224, SHA-256, SHA-384, SHA-512**
- Block chaining modes: **CBC, XTS**
- IV-Handling mechanisms:
  - **CBC: essiv**
  - **XTS: plain or plain64**

and cryptographic key sizes as allowed by the cipher specifications that meet the following: **LUKS-based dm-crypt Linux partition encryption schema**.

6.3 User Data Protection (FDP)

6.3.1 Subset Access Control Policy (FDP_ACC.1)

FDP_ACC.1.1 The TSF shall enforce the **Discretionary Access Control Policy** on **processes acting on behalf of users as subjects and file system objects (ordinary files, directories, symbolic links, device special files, UNIX Domain socket special files, named pipes), IPC objects (message queues, semaphores, shared memory segments) and all operations among subjects and objects covered by the DAC policy**.

6.3.2 Security Attribute Based Access Control (FDP_ACF.1)

FDP_ACF.1.1 The TSF shall enforce the **Discretionary Access Control Policy** to objects based on the following:

- **The filesystem user identity and group membership(s) associated with a subject; and**
- **The following access control attributes associated with an object:**
  - **File system objects:**

    POSIX ACLs and permission bits.

    (ACLs can be used to grant or deny access to the granularity of a single user or group using Access Control Entries. Those ACL entries

include the standard Unix permission bits. Posix ACLs can be used for file system objects within the ext3 and ext4 file system).

Access rights for filesystem objects are:

- **read**
- **write**
- **execute (ordinary files)**
- **search (directories)**

- **IPC objects:**

  permission bits

  Access rights for IPC objects are:

  - **read**
  - **write**

FDP_ACF.1.2 The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed:

File system objects within the ext3 and ext4 file systems:

A subject must have search permission for every element of the pathname and the requested access for the object. A subject has a specific type access to an object if:

- **The subject has been granted access according to the ACL_USER_OBJ or ACL_OTHER type entry in the ACL of the object**

Or

- **The subject has been granted access by an ACL_USER, ACL_GROUP_OBJ or ACL_GROUP entry and the associated right is also granted by the ACL_MASK entry of the ACL if the ACL_MASK entry exist**

Or

- **The subject has been granted access by the ACL_GROUP_OBJ entry and no ACL_MASK entry exists in the ACL of the object.**

File system objects in other file systems:

A subject must have search permission for every element of the pathname and the requested access for the object. A subject has a specific type access to an object if:

- **The subject has the file system userid of the owner of the object and the requested type of access is within the permission bits defined for the owner**

Or

- **The subject has not the file system userid of the owner of the object but the file system group id identical to the file system objects group id and the requested type of access is within the permission bits defined for the group**

  Or

- **The subject has neither the file system userid of the owner of the object nor is the file system group id identical to the file system object group id and requested type of access is within the permission bits defined for "world"**

IPC objects:

Access permissions are defined by permission bits of the IPC object. The process creating the object defines the creator, owner and group based on the userid of the current process. Access of a process to an IPC object is allowed, if

- **the effective userid of the current process is equal to the userid of the IPC object creator or owner and the „owner" permission bit for the requested type of access is set or**
- **the effective userid of the current process is not equal to the userid of the IPC object creator or owner and the effective group id of the current process is equal to the group id of the IPC object and the "group" permission bit for the requested type of access is set or**
- **The "world" permission bit for the requested type of access is set for users that do not satisfy one of the first two conditions.**

FDP_ACF.1.3 The TSF shall explicitly authorize access of subjects to objects based on the following additional rules:

File System Objects:

A process with a user ID of 0 is known as a root user process. These processes are generally allowed all access permissions. But if a root user process requests execute permission for a program (as a file system object), access is granted only if execute permission is granted to at least one user.

IPC objects:

A process with a user ID of 0 is known as a root user process. These processes are generally allowed all access permissions.

FDP_ACF.1.4 The TSF shall explicitly deny access of subjects to objects based on the following additional rules:

Write access to file system objects other than device special files on a file system mounted as read-only is always denied.

Write access to a file marked as immutable is always denied.

6.3.3 Full Residual Information Protection (FDP_RIP.2)

FDP_RIP.2.1 The TSF shall ensure that any previous information content of a resource is made unavailable upon the **allocation of the resource to** all objects.

6.3.4 Basic data exchange confidentiality (FDP_UCT.1)

FDP_UCT.1.1 **Refinement:** The TSF shall enforce the **Discretionary Access Control Policy** to **be able to transmit and receive** ~~user data~~ **objects** in a manner protected from unauthorized disclosure.

Application Note: *Confidentiality of data during transmission is ensured when the secured protocol ssh is used. User processes are still bound by the discretionary access control policy with respect to the data they are able to transfer. The TOE is able to act both as a server and a client for ssh connections.*

6.3.5 Data exchange integrity (FDP_UIT.1)

FDP_UIT.1.1 The TSF shall enforce the **Discretionary Access Control Policy** to **transmit and receive** user data in a manner protected from **modification and insertion** errors.

FDP_UIT.1.2 The TSF shall be able to determine on receipt of user data, whether modification or insertion has occurred.

Application Note: *Integrity of data during transmission is ensured when the secured protocol ssh is used. User processes are still bound by the discretionary access control policy with respect to the data they are able to transfer. The TOE is able act both as a server and a client for ssh connections.*

6.4 IDENTIFICATION AND AUTHENTICATION (FIA)

6.4.1 Authentication Failure Handling (FIA_AFL.1)

FIA_AFL.1.1 The TSF shall detect when an authorized administrator configurable positive integer of consecutive unsuccessful authentication attempts occur related to any authorized user authentication process.

FIA_AFL.1.2 When the defined number of consecutive unsuccessful authentication attempts has been met, the TSF shall:

- For all administrator accounts, "disable" the account for an authorized administrator configurable time period such that there can be no more than a configurable attempt per minute.
- For all other accounts, disable the user logon account until it is re-enabled by the authorized administrator.
- For all disabled accounts, any response to an authentication attempt given to the user shall not be based on the result of that authentication attempt.

Application Note:
The configuration of this functional aspect is done by modifying the parameters for the PAM modules in the PAM configuration files stored in /etc/pam.d/. By default, the number of consecutive unsuccessful authentication attempts before the account will be suspended (for authorized administrators) or disabled (for authorized users) are set to 10 for authorized administrator and to 5 for other users.

6.4.2 User Attribute Definition (FIA_ATD.1)

FIA_ATD.1.1 The TSF shall maintain the following list of security attributes belonging to individual users:

- **user identifier;**
- **group memberships;**
- **authentication data;**
- **security-relevant roles.**

Application Note: *The maintenance of the security relevant role to an user is done implicitly with the maintenance of the group membership for each user.*

Application Note: *Authentication data refer both to user password and software token that can be used for subsequent identification and authentication with the TSF or other remote IT systems. Please see the application note for FIA_UAU.5 for a list of token-based authentication mechanisms and their associated tokens.*

6.4.3 Verification of Secrets (FIA_SOS.1)

FIA_SOS.1.1 The TSF shall provide a mechanism to verify that secrets meet **the following:**

I. For password-based authentication methods during local or remote user's login

a. Password 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;

b. Password is not reused within the last administrator-settable (5) number of passwords used by that user;

c. Password is not easily-guessable (e.g. it does not contain user name, user account, etc., and it is not based on a dictionary word);

d. Password does not contain more than 2 equal numbers or letters or symbols consecutives;

e. Password is at least 8 characters different from the current password.

II. For LUKS-formatted devices

a. Passphrase is at least 16 characters in length, consisting of at least two upper and two lower case letter, at least two number, and two symbol;

b. Passphrase is not easily-guessable (e.g. it does not contain user name, surname, user account, etc.., and it is not based on a dictionary word);

c. Passphrase does not contain more than two equal numbers or letters or symbols consecutives.

III. For dual-factor authentication methods

a. DSA keys meet the signature authentication defined by U.S. NIST FIPS PUB 186-4 [FIPS186-4] for all specified cryptographic key sizes:

- **1024 bits;**
- **2048 bits;**
- **3072 bits;**

b. RSA keys meet the signature authentication defined by U.S. NIST FIPS PUB 186-2 [FIPS186-2] for all specified cryptographic key sizes:

- **1024 bits;**
- **1536 bits;**
- **2048 bits;**
- **3072 bits;**
- **4096 bits.**

IV. For public-key-based authentication methods

a. DSA keys meet the format of ssh-dss for "publickey" authentication defined by RFC 4252 and specified cryptographic key sizes:

- **1024 bits;**
- **2048 bits;**
- **3072 bits;**

b. RSA keys meet the format of ssh-rsa for "publickey" authentication defined by RFC 4252 and specified cryptographic key sizes:

- **1024 bits;**
- **1536 bits;**
- **2048 bits;**
- **3072 bits;**
- **4096 bits.**

Application Note: *For password-based authentication methods:*

- *The TSF provides a mechanism for the authorized administrator to set the password complexity and a password history such that a user cannot reuse any password that is on the password history list;*
- *The TOE password change is implemented using the PAM library. The specification of the quality of new passwords is in charge to the pam_cracklib.so PAM module.*

Application Note: *For dual-factor authentication methods, the authentication is implemented using the PAM library and the OpenSSL library. The overall authentication process is in charge to to the pam_finxse_dfa.so PAM module.*

Application Note: *For public-key-based authentication methods when initiating a communication over a trusted channel, the evaluation of the RSA and DSA keys used for the SSH protocol is performed by OpenSSH.*

6.4.4 Authentication (FIA_UAU.2)

FIA_UAU.2.1 The TSF shall require each user to be successfully authenticated before allowing any other TSF-mediated actions on behalf of that user.

Application Note: *Untrusted processes running on behalf of a normal user may use network functions to import and export data they have access to. This process may therefore export user data without authenticating or even knowing the identity of a user receiving such data. This is not considered to be a violation of the security policy with respect to identification and authentication and discretionary access control, since it is well-known that discretionary access control cannot control flow of information. An example of such an export function is a user process running a web-server on an unprivileged port. Still this process is limited in its access by the security policy of the TOE.*

6.4.5 Multiple authentication mechanisms (FIA_UAU.5)

FIA_UAU.5.1 The TSF shall provide the following authentication mechanisms:

a. **Passoword-based authentication, based on username and password;**

b. **Dual-factor authentication, based on software token verification data;**

c. **Public-key-based authentication, based on software verification data.**

to support user authentication.

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

a. **Authentication based on username and password is performed for TOE-originated requests and credentials stored by the TSF;**

b. **Authentication based on software token verification data is performed for TOE-originated requests. If dual-factor authentication is enabled for the user, it requires a successful password-based authentication first so that both authentication methods must succeed to get the user authenticated;**

c. **For SSH, both, the password-based and public-key-based authentication methods can be enabled at the same time. In this case, the public-key-based authentication method is tried before the password-based authentication. If the public-key-based authentication succeeds, the user is authenticated. If the public-key-based authentication fails, the password-based authentication is applied. If the password-based authentication fails, the user login request is denied.**

Application Note: *For public-key-based authentication methods, the TOE is able to maintain the following types of software verification data:*

- *SSH user keys: The TOE as server part is able to store the public part of the SSH user key for the user account the user wants to access.*

Application Note: *For dual-factor authentication methods, the TOE is able to maintain the following types of software tokens and their verification data:*

- *RSA key pairs as defined by U.S. NIST FIPS PUB 186-2 [FIPS186-2]*
- *DSA key pairs as defined by U.S. NIST FIPS PUB 186-4 [FIPS186-4]*

6.4.6 Re-authenticating (FIA_UAU.6)

FIA_UAU.6.1 **Refinement:** The TSF shall re-authenticate the user ~~under the conditions~~ **when changing authentication data.**

Application Note: *If the TOE is requiring the user to change authentication data upon having just authenticated (e.g., initial logon, session unlock), the user is considered to be re-authenticated. The re-authentication mechanism does not apply to changes on software tokens.*

6.4.7 Protected Authentication Feedback (FIA_UAU.7)

FIA_UAU.7.1 The TSF shall provide only **obscured feedback** to the user while the authentication is in progress.

6.4.8 Identification (FIA_UID.2)

FIA_UID.2.1 **Refinement:** The TSF shall require each user to ~~be successfully identified~~ **identify himself** before allowing any other TSF-mediated actions on behalf of that user.

6.4.9 User-Subject Binding (FIA_USB.1)

FIA_USB.1.1 The TSF shall associate the following user security attributes with subjects acting on behalf of that user: **the security attributes identified in FIA_ATD.1.**

FIA_USB.1.2 The TSF shall enforce the following rules on the initial association of user security attributes with subjects acting on behalf of users:

- **Upon successful identification and authentication, the login UID, the real UID, the filesystem UID and the effective UID shall be those specified in the user entry for the user that has authenticated successfully,**
- **Upon successful identification and authentication, the real GID, the filesystem GID and the effective GID shall be those specified via the primary group membership attribute in the user entry,**
- **Upon successful identification and authentication, the supplemental GIDs shall be those specified via the supplemental group membership assignment for the user entry,**
- **The role associated with a subject shall be one of the authorized roles assigned to the user.**

FIA_USB.1.3 The TSF shall enforce the following rules governing changes to the user security attributes associated with subjects acting on behalf of users:

- **The effective and filesystem UID of a subject can be changed by the use of an executable with the SETUID bit set. In this case the program is executed with the effective and filesystem UID of the owning UID of the file storing the program. These newly set effective and filesystem UIDs are used for the DAC permission validation. The real and login UID remain unchanged.**
- **The effective and filesystem GID of a subject can be changed by the use of an executable with the SETGID bit set. In this case the program is executed with the effective and filesystem GID of the owning GID of the file storing the program. These newly set effective and filesystem GIDs**

are used for the DAC permission validation. The real GID remains unchanged.

- **The real, effective and filesystem UID of a subject can be changed by the use of the set*uid system call family for the calling application. These system calls are restricted to the root user.**
- **The real, effective and filesystem GID of a subject can be changed by the use of the set*gid system call family for the calling application. These system calls are restricted to the root user.**
- **The set of supplemental GIDs of a subject can be changed by the use of the set groups system call for the calling application. This system call is restricted to the root user.**

6.5 SECURITY MANAGEMENT (FMT)

6.5.1 Management of Security Functions Behaviour (for specification of auditable events) (FMT_MOF.1(1))

FMT_MOF.1(1).1 **Refinement:** The TSF shall restrict the ability to **disable and enable** the **audit** functions **and to specify which events are to be audited (see FAU_SEL.1.1)** to **the authorized administrators**.

Application Note: *To "specify" means the ability to select what events will be audited.*

6.5.2 Management of Security Functions Behaviour (for authentication data) (FMT_MOF.1(2))

FMT_MOF.1(2).1 **Refinement**: The TSF shall restrict the ability to **manage the values of security attributes associated with user authentication data** to **authorized administrators**.

Application Note: *The word "manage" includes but is not limited to create, initialize, change default, modify, delete, clear, append, and query. The security attributes associated with user authentication data referenced by this requirement include those that are specified by FIA_AFL and FIA_SOS.*

6.5.3 Management of Object Security Attributes (FMT_MSA.1)

FMT_MSA.1.1 **Refinement:** The TSF shall enforce the **Discretionary Access Control Policy** to restrict the ability to **modify** the ~~security~~ **access control** attributes **and the owner associated with a named object** to **the authorized administrators and to the original creator.**

6.5.4 Secure Security Attributes (FMT_MSA.2)

FMT_MSA.2.1 **Refinement:** The TSF shall ensure that only ~~secure~~ **valid** values are accepted for **security attributes.**

Application Note: *"Valid" implies that the values assigned to security attributes are valid with respect to the secure state and fall within an appropriate range for that attribute.*

6.5.5 Static Attributes Initialization (FMT_MSA.3)

FMT_MSA.3.1 The TSF shall enforce the **Discretionary Access Control policy** to provide **restrictive** default values for security attributes that are used to enforce the SFP.

FMT_MSA.3.2 The TSF shall allow the **authorized administrator** to specify alternative initial values to override the default values when an object or information is created.

Application Note: *The default value for permission bits is specified with the umask value which specifies the permission bits for newly created objects. This value has an initial setting of 022 or the value specified in /etc/login.defs. Only authorized administrator can change that initial value. Users can change their umask value at any time. For ACLs, the default ACL is provided for the root directory which, in case of absence of a default ACL entry is consistent with the umask.*

6.5.6 Management of TSF Data (for general TSF data) (FMT_MTD.1(1))

FMT_MTD.1(1).1 The TSF shall restrict the ability to **manage** the **TSF data except for audit records, user security attributes, authentication data, and critical cryptographic security parameters** to **authorized administrators.**

Application Note: *The word "manage" includes but is not limited to create, initialize, change default, modify, delete, clear, append, and query. Security attributes associated with user authentication data include password length, password expiration, password history, etc. The restrictions for audit records, user security attributes, authentication data, and critical cryptographic security parameters are specified below.*

6.5.7 Management of TSF Data (for audit data) (FMT_MTD.1(2))

FMT_MTD.1(2).1 The TSF shall restrict the ability to **query, delete, and clear** the **audit records** to **authorized administrators.**

Application Note: *This requirement applies to actions taken on the entire audit file/log, not actions on individual audit records*.

6.5.8 Management of TSF Data (for initialization of user security attributes) (FMT_MTD.1(3))

FMT_MTD.1(3).1 The TSF shall restrict the ability to **initialize** the **user security attributes** to **authorized administrators.**

6.5.9 Management of TSF Data (for modification of user security attributes, other than authentication data) (FMT_MTD.1(4))

FMT_MTD.1(4).1 The TSF shall restrict the ability to **modify** the **user security attributes, other than authentication data,** to **authorized administrators**.

6.5.10 Management of TSF Data (for modification of authentication data) (FMT_MTD.1(5))

FMT_MTD.1(5).1 The TSF shall restrict the ability to **modify** the **authentication data** to **authorized users modifying their own authentication data.**

6.5.11Management of TSF Data (for reading of authentication data) (FMT_MTD.1(6))

FMT_MTD.1(6).1 **Refinement:** The TSF shall ~~restrict the ability to~~ **prevent reading** ~~the~~ **of authentication data** ~~to~~**.**

6.5.12Management of TSF Data (for critical cryptographic security parameters) (FMT_MTD.1(7))

FMT_MTD.1(7).1 The TSF shall restrict the ability to **manage** the **critical cryptographic security parameters and data related to cryptographic configuration** to **authorized administrators.**

Application Note: *The word "manage" includes but is not limited to create, initialize, change default, modify, delete, clear, append, and query.*

6.5.13Revocation (to authorized administrators) (FMT_REV.1(1))

FMT_REV.1(1).1 The TSF shall restrict the ability to revoke **security attributes** associated with the **users** under the control of the TSF to **authorized administrators**.

Application Note: *The phrase "revoke security attributes" means to change attributes so that access is revoked.*

FMT_REV.1(1).2 **Refinement:** The TSF shall enforce the ~~rules~~ **revocation of security-relevant authorizations at the next logon**.

Application Note: *Security-relevant authorizations include the ability of authorized users to log in or perform privileged operations. An example of revoking a security-relevant authorization is the deletion of a user account upon which TOE access is immediately terminated.*

6.5.14Revocation (to owners and authorized administrators) (FMT_REV.1(2))

FMT_REV.1(2).1 **Refinement:** The TSF shall restrict the ability to revoke **security attributes** ~~associated with the~~ **of objects** ~~under the control of the TSF~~ to **owners of the object and authorized administrators.**

Application Note: *The term "revoke security attributes" means "change attributes so that access is revoked".*

FMT_REV.1(2).2 **Refinement:** The TSF shall enforce the ~~rules~~ **revocation of access rights associated with objects when an access check is made.**

Application Note: *The state where access checks are made determines when the access control policy enforces revocation. The access control policy may include immediate or delayed revocation. The access rights are considered to have been revoked when all subsequent access control decisions made by the TSF use the new access control information. In cases where a previous access control decision was made to permit an operation, it is not required that every subsequent operation make an explicit access control decision.*

6.5.15Time-limited authorization (FMT_SAE.1)

FMT_SAE.1.1 The TSF shall restrict the capability to specify an expiration time for **user authentication data** to **the authorized administrator**.

FMT_SAE.1.2 **Refinement:** ~~For each of these security attributes,~~ The TSF shall be able to **force the associated authorized user to change his authentication information prior to being**

able to successfully log on after the expiration time ~~for the indicated security attribute~~ has passed.

Application Note: *Expiration time for user authentication data does not apply to dual-factor authentication method.*

6.5.16Specification of Management Functions (FMT_SMF.1)

FMT_SMF.1.1 **Refinement:** The TSF shall be capable of performing the following **security** management functions:

- **Management of auditing;**
- **Management of cryptographic services;**
- **Management of the access control policy (DAC);**
- **Management of identification and authentication policy;**
- **Management of user security attributes.**

6.5.17Security Roles (FMT_SMR.1)

FMT_SMR.1.1 The TSF shall maintain the roles:

- **Authorized administrator;**
- **Authorized non-administrative user.**

FMT_SMR.1.2 The TSF shall be able to associate users with roles.

Application Note: *Administrative actions can only be performed when the calling subject possesses the effective UID or file system UID of zero (also called the root user). As the account for the root user is disabled for direct logon, authorized administrators are defined as users who are assigned to the "admins" group. This group allows the use of the "su" application which is the only way to assume the root user capabilities.*
Authorized non-administrative users are authorized by the Discretionary Access Control Policy to modify their own object security attributes and their own authentication data.

6.6 PROTECTION OF THE TOE SECURITY FUNCTIONS (FPT)

6.6.1 Reliable Time Stamps (FPT_STM.1)

FPT_STM.1.1 The TSF shall be able to provide reliable time stamps.

6.6.2 TSF Self Test (FPT_TST.1)

FPT_TST.1.1 The TSF shall run a suite of self tests **at the request of the authorized administrators** to demonstrate the correct operation of **the TSF**.

FPT_TST.1.2 **Refinement**: The TSF shall provide authorized ~~users~~ **administrators** with the capability to verify the integrity of **TSF data (with the exception of audit data).**

FPT_TST.1.3 **Refinement**: The TSF shall provide authorized ~~users~~ **administrators** with the capability to verify the integrity of **stored TSF executable code.**

6.6.3 Failure with preservation of secure state - full buffer overflow protection (FPT_FLS.1(1))

FPT_FLS.1(1).1 The TSF shall preserve a secure state when the following types of failures occur:

1) Execution of code on a process' or thread's stack;

2) Modification of a function return address on the process' or thread's stack to jump to previously known processor instructions;

3) Modification of process section other than the segment that holds compile time initialized data and segment holding the mapping of all uninitialized variables.

for the runtime instances of the following binaries:

a) all user-provided applications and their depending libraries that are compiled and linked with the following properties:

i) presence of the ELF program header entry of PT_GNU_STACK and the absense of the PF_X bit in the p_flags ELF header flags;

ii) presence of the ELF program header entry of PT_GNU_RELRO with the memory range information covering the following ELF sections: .tdata, .preinit_array, .init_array, .fini_array, .ctors, .dtors, .jcr, .data.rel.ro, .dynamic, .got including .got.plt.

Application Note: *The secure state implied with this functionality covers the following aspects where the following list explains the implication of each bullet above:*

a) When exploiting buffer overruns of an application, the attacker cannot feed code onto the stack and execute it.

b) When exploiting buffer overruns of an application, the modify of the return address stored on a stack to jump to a previously known code segment is much harder to achieve by an attacker. Due to the address randomization, any memory address of code already present with the application or loaded libraries will be different with each startup of the application.

c) The ELF header sections listed above are set read-only using the mprotect system call by the loader before the application gains control. When exploiting buffer overruns, the attacker cannot modify information in those memory sections. These sections store offset tables required for the dynamic linking mechanism and, if abused, allow attackers to modify the jump addresses of object accesses. Full protection against this type of attack can only be achieved if the application and all depending shared libraries are compiled linked with full protection enabled. When at least one shared library the application depends on or the application itself is compiled and linked with partial protection (see FPT_FLS.1(2)), only partial protection against this type of attack is available for the given application.

Application Note: *To enforce the functionality in bullet b), /proc/sys/kernel/randomize_va_space must contain a 2 (1 implies that the address randomization is enabled except for the brk systemcall).*

Application Note: *During standard compilation of applications, the stack execution protection is enabled. To ensure the presence of the PT_GNU_STACK ELF program header entry and the absence of the PF_X bit in the p_flags ELF header flags, the following considerations must be applied by a programmer as any of the following operations disable the stack execution protection:*

- *The following linker option must not be used: "-z execstack" (gcc: "-Wl,-z,execstack").*
- *The following assembler option must not be used: "--execstack" (gcc: "-Wa,--execstack").*
- *Modifications of an ELF program header entry in an already compiled binary which change the PT_GNU_STACK and PF_X flags (like using the execstack(8) application) must not be performed.*

- *The application or library code must not contain trampolines such as nested functions pushed onto the stack which passed as pointers to functions as this would also enable the stack execution support.*

Application Note: *To ensure the presence of PT_GNU_RELRO covering the proper ELF sections, the application must be linked with the provided linker using the linker options of "-z relro -z now" (using the GCC compiler using the compiler options of "-Wl,-z,relro,-z,now" can be used which are passed to the linker). In addition, an application must be compiled as PIE with the gcc option of "-fPIE". Contrary, a shared library must be compiled as PIC with the gcc option of "-fPIC".*

6.6.4 Failure with preservation of secure state - partial buffer overflow protection (FPT_FLS.1(2))

FPT_FLS.1(2).1 The TSF shall preserve a secure state when the following types of failures occur:

1) Execution of code on a process' or thread's stack;

2) Modification of a function return address on the process' or thread's stack to jump to previously known processor instructions;

3) Modification of process sections other than the segment that holds compile time initialized data, the segment holding the mapping of all uninitialized variables, and the procedure linking table (PLT).

for:

a) all libraries in the TSF;

b) all user-provided applications and their depending libraries that are compiled and linked with the following properties:

i) presence of the ELF program header entry of PT_GNU_STACK and the absense of the PT_X bit in the p_flags ELF header flags;

ii) presence of the ELF program header entry of PT_GNU_RELRO with the memory range information covering the following ELF sections: .tdata, .preinit_array, .init_array, .fini_array, .ctors, .dtors, .jcr, .data.rel.ro, .dynamic, .got excluding .got.plt.

Application Note: *The only difference between full and partial RELRO is that in partial RELRO the .got.plt ELF section is left unprotected and is therefore read/writable. This difference allows lazy bindings during the dynamic linking process.*

Application Note: *All application notes from FPT_FLS.1(1) apply except the following.*

Application Note: *To ensure the presence of PT_GNU_RELRO covering the proper ELF sections, the application must be linked with the provided linker using the linker options of "-z relro" (using the GCC compiler using the compiler options of "-Wl,-z,relro" can be used which are passed to the linker). In addition, an application must be compiled as PIE with the gcc option of "-fPIE". Contrary, a shared library must be compiled as PIC with the gcc option of "-fPIC".*

6.7 TOE Access (FTA)

6.7.1 User-Initiated Locking (FTA_SSL.2)

FTA_SSL.2.1 The TSF shall allow user-initiated locking of the user's own interactive session by:

- clearing or overwriting display devices, making the current contents unreadable.

- disabling any activity of the user's data access/display devices other than unlocking the session.

FTA_SSL.2.2 **Refinement:** The TSF shall require the ~~following events to occur~~ **user to re-authenticate** using **password-based authentication** ~~prior~~ to ~~unlocking~~ **unlock** the session.

Application note: *Also administrative users can lock/unlock their interactive session.*

6.7.2 Default TOE Access Banners (FTA_TAB.1)

FTA_TAB.1.1 **Refinement:** Before establishing a user session, the TSF shall display an **authorized-administrator specified** advisory **notice and consent** warning message regarding unauthorized use of the TOE.

Application Note: *It should be noted that not all interactions with the TSF will require an access banner to be displayed. The requirement should be interpreted to cover only sessions initiated by a human. It shall be possible to enforce the advisory notice and consent so that the user is forced to explicitily accept all terms stated in the access banner to be able entering a new session.*

6.7.3 TOE Access History (FTA_TAH.1)

FTA_TAH.1.1 **Refinement:** Upon successful **interactive** session establishment, the TSF shall display **to the authorized user** the **date, time, and location** of ~~the~~ **that authorized user's** last successful **interactive** session establishment ~~to the user~~.

FTA_TAH.1.2 **Refinement:** Upon successful **interactive** session establishment, the TSF shall display **to the authorized user** the **date, time, and location** of the last unsuccessful attempt ~~to session establishment~~ and the number of unsuccessful attempts **at interactive session establishment for that user identifier** since the last successful **interactive** session establishment.

FTA_TAH.1.3 **Refinement:** The TSF shall not erase the access history information from the **authorized** user interface without giving the **authorized** user ~~an~~ the opportunity to review the information.

6.8 TRUSTED PATH/CHANNELS (FTP)

6.8.1 Inter-TSF trusted channel (FTP_ITC.1)

FTP_ITC.1.1 **Refinement:** The TSF shall provide a communication channel between itself and another **remote** trusted IT product that is logically distinct from other communication channels and provides assured identification of its end points and protection of the channel data from modification or disclosure.

Application Note: *The communication channel is cryptographically-protected using SSH protocol version 2.*

FTP_ITC.1.2 The TSF shall permit **another trusted IT product** to initiate communication via the trusted channel.

FTP_ITC.1.3 The TSF shall initiate communication via the trusted channel for **all security functions specified in the ST that interact with remote trusted IT systems**.

Application Note: *The SSH protocol implements a bi-directional authentication mechanism as follows:*

- Server-side authentication: the user identification and authentication via user name and password / SSH user key allows the server to authenticate the client.
- Client-side authentication: the SSH host key verification performed by the SSH client during each connection attempt allows the client to authenticate the server.

6.9 SECURITY FUNCTIONAL REQUIREMENTS RATIONALE

6.9.1 Security requirements coverage

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

Security Functional Requirements	Objectives
Security Audit (FAU)	
Audit Data Generation (FAU_GEN.1)	O.AUDIT_GENERATION
User Identity Association (FAU_GEN.2)	O.AUDIT_GENERATION
Audit Review (FAU_SAR.1)	O.AUDIT_REVIEW
Restricted Audit Review (FAU_SAR.2)	O.AUDIT_PROTECTION
Selectable Audit Review (FAU_SAR.3)	O.AUDIT_REVIEW
Selective Audit (FAU_SEL.1)	O.AUDIT_GENERATION
Protected Audit Trail Storage (FAU_STG.1) Action in case of possible audit data loss (FAU_STG.3) Prevention of audit data loss (FAU_STG.4)	O.AUDIT_PROTECTION
Cryptographic Support (FCS)	
Cryptographic key generation (Symmetric Keys) (FCS_CKM.1(1))	O.CRYPTO_NET
Cryptographic key generation (MEDIA) (FCS_CKM.1(2))	O.CRYPTO_MEDIA O.USER_AUTHENTICATION
Cryptographic key generation (RSA) (FCS_CKM.1(3))	O.CRYPTO_NET O.USER_AUTHENTICATION
Cryptographic key generation (DSA) (FCS_CKM.1(4))	O.CRYPTO_NET O.USER_AUTHENTICATION
Cryptographic key distribution (FCS_CKM.2)	O.CRYPTO_NET
Cryptographic key destruction (FCS_CKM.4_EXT)	O.CRYPTO_NET

Security Functional Requirements	Objectives
Cryptographic operation (FCS_COP.1(1))	O.CRYPTO_NET O.USER_AUTHENTICATION
Cryptographic Operation (LUKS) (FCS_COP.1(2))	O.CRYPTO_MEDIA
Authentication (FIA_UAU.2)	O.CRYPTO_MEDIA O.CRYPTO_NET O.USER_AUTHENTICATION
Multiple authentication mechanisms (FIA_UAU.5)	O.CRYPTO_MEDIA O.CRYPTO_NET O.USER_AUTHENTICATION
User Data Protection (FDP)	
Subset Access Control Policy (FDP_ACC.1)	O.ACCESS, O.DISCRETIONARY_ACCESS, O.PROTECT
Security Attribute Based Access Control (File System Objects) (FDP_ACF.1)	O.ACCESS, O.DISCRETIONARY_ACCESS, O.PROTECT
Security Attribute Based Access Control (IPC Objects) (FDP_ACF.1)	O.ACCESS, O.DISCRETIONARY_ACCESS, O.PROTECT
Full Residual Information Protection (FDP_RIP.2)	O.PROTECT, O.RESIDUAL_INFORMATION
Basic data exchange confidentiality (FDP_UCT.1)	O.TRUSTED_CHANNEL
Data exchange integrity (FDP_UIT.1)	O.TRUSTED_CHANNEL
Identification and Authentication (FIA)	
Authentication Failure Handling (FIA_AFL.1)	O.ACCESS
User Attribute Definition (FIA_ATD.1)	O.ACCESS
Verification of Secrets (FIA_SOS.1)	O.PROTECT, O.USER_AUTHENTICATION

Security Functional Requirements	Objectives
Authentication (FIA_UAU.2)	O.USER_AUTHENTICATION
Multiple authentication mechanisms (FIA_UAU.5)	O.USER_AUTHENTICATION
Re-authenticating (FIA_UAU.6)	O.USER_AUTHENTICATION
Protected Authentication Feedback (FIA_UAU.7)	O.PROTECT
Identification (FIA_UID.2)	O.USER_IDENTIFICATION
User-Subject Binding (FIA_USB.1)	O.AUDIT_GENERATION, O.DISCRETIONARY_ACCESS
Security Management (FMT)	
Management of Security Functions Behavior (for specification of auditable events) (FMT_MOF.1(1))	O.MANAGE
Management of Security Functions Behavior (for authentication data) (FMT_MOF.1(2))	O.MANAGE
Management of Object Security Attributes (FMT_MSA.1)	O.MANAGE, O.DISCRETIONARY_USER_CONTROL
Secure Security Attributes (FMT_MSA.2)	O.PROTECT
Static Attributes Initialization (FMT_MSA.3)	O.DISCRETIONARY_ACCESS, O.MANAGE
Management of TSF Data (for general TSF data) (FMT_MTD.1(1))	O.MANAGE
Management of TSF Data (for audit data) (FMT_MTD.1(2))	O.MANAGE
Management of TSF Data (for initialization of user security attributes) (FMT_MTD.1(3))	O.MANAGE
Management of TSF Data (for modification of user security attributes, other than authentication data) (FMT_MTD.1(4))	O.MANAGE
Management of TSF Data (for modification of authentication data) (FMT_MTD.1(5))	O.MANAGE

Security Functional Requirements	Objectives
Management of TSF Data (for reading of authentication data) (FMT_MTD.1(6))	O.PROTECT
Management of TSF Data (for critical cryptographic security parameters) (FMT_MTD.1(7))	O.MANAGE
Revocation (to authorized administrators) (FMT_REV.1(1))	O.MANAGE O.ACCESS
Revocation (to owners and authorized administrators) (FMT_REV.1(2))	O.MANAGE, O.ACCESS, O.DISCRETIONARY_USER_CONTROL, O.PROTECT
Time-limited authorization (FMT_SAE.1)	O.MANAGE
Specification of Management Functions (FMT_SMF.1)	O.MANAGE
Security Roles (FMT_SMR.1)	O.MANAGE
Protection of the TOE Security Functions (FPT)	
Reliable Time Stamps (FPT_STM.1)	O.AUDIT_GENERATION
TSF Testing (FPT_TST.1)	O.ENFORCEMENT
Failure with preservation of secure state - full buffer overflow protection (FPT_FLS.1(1))	O.RUNTIME.PROTECTION
Failure with preservation of secure state - partial buffer overflow protection (FPT_FLS.1(2))	O.RUNTIME.PROTECTION
TOE Access (FTA)	
User-Initiated Locking (FTA_SSL.2)	O.ACCESS, O.USER_AUTHENTICATION
Default TOE Access Banners (FTA_TAB.1)	O.DISPLAY_BANNER
TOE Access History (FTA_TAH.1)	O.ACCESS_HISTORY
Trusted path/channels (FTP)	
(Inter-TSF trusted channel) FTP_ITC.1	O.TRUSTED_CHANNEL

Table 8: Mapping of security functional requirements to security objectives

6.9.2 Security requirements sufficiency

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

Objectives	Rationale
O.ACCESS	The TOE must protect itself and the resources it controls from unauthorized access. FDP_ACC.1 enforces the Discretionary Access Control (DAC) policy on all subjects and all named objects identified and all operations among them and covered by DAC policy. The DAC policy specifies the access rules between all subjects and all named objects controlled by the TOE. While authorized users are trusted to some extent, this requirement ensures only authorized access is allowed to named objects. FDP_ACF.1 specifies the DAC policy rules that will be enforced by the TSF to File System Objects and IPC Objects. It determines if an operation among subjects and named objects is allowed. Furthermore, it specifies the rules to explicitly authorize or deny access to a named object based upon security attributes. FIA_AFL.1 provides a detection mechanism for unsuccessful authentication attempts. The requirement enables an authorized administrator to configure a threshold that prevents unauthorized users from gaining access to authorized user's account by guessing authentication data. This mechanism prevents access disabling the targeted account thus limiting an unauthorized user's ability to gain unauthorized access to the TOE. FIA_ATD.1 defines the attributes of users, including a userid that is used by the TOE to determine a user's identity and enforce what type of access the user has to the TOE (e.g., the TOE associates a userid with any role(s) they may assume). FMT_REV.1(1) ensures that the authorized administrator has the ability to revoke security attributes to a specific user. This revocation is immediate and helps authorized administrators control the ability of authorized users to log in or perform privileged operations. FMT_REV.1(2) ensures that the authorized administrator and owners of objects have the ability to revoke security attributes to a specific user. This

	revocation occurs when an access check is made and helps authorized administrators and owners control the ability of users accessing named objects. FTA_SSL.2 is used to prevent unauthorized access to the TOE and its resources when an interactive session is left unattended. It enables the authorized user to lock his interactive session before leaving the session unattended. This eliminates any chance for any user to acquire unauthorized access to an unattended session because there is no time interval of inactivity before the session is locked. The authorized user needs to re-authenticate to unlock his session.
O.ACCESS_HISTORY	FTA_TAH.1 is used to provide information about previous interactive sessions (i.e., date and time). This information is displayed to the authorized user upon each successful interactive session establishment. This requirement gives the authorized users the ability to verify their last successful interactive session and thus, is a means for determining if the previous successful interactive session establishment was authorized or not.
O.AUDIT_GENERATION	FAU_GEN.1 defines the set of events that the TOE must be capable of recording. This requirement ensures that the authorized administrator has the ability to audit any security relevant event that takes place in the TOE. This requirement also defines the information that must be contained in the audit record for each auditable event. There is a minimum of information that must be present in every audit record and this requirement defines that, as well as the additional information that must be recorded for each auditable event. This requirement also places a requirement on the level of detail that is recorded on any additional security functional requirements. FAU_GEN.2 ensures that the audit records associate a user identity with the auditable event. The association is accomplished using the userid of the authorized user. FAU_SEL.1 allows the authorized administrator to configure which auditable events will be recorded in the audit trail. This provides the authorized

	administrator with the flexibility in recording only those events that are deemed necessary by site policy, thus reducing the amount of resources consumed by the audit mechanism. FIA_USB.1 plays a role is satisfying this objective by requiring a binding of security attributes associated with users that are authenticated with the subjects that represent them in the TOE. This only applies to authenticated users, since the identity of unauthenticated users cannot be confirmed. Therefore, the audit trail may not always have the proper identity of the user that causes an audit record to be generated (e.g., an attacker/user providing another user's user identifier). FPT_STM.1 ensures that the time stamps used to create the audit records are reliable. The time and date included in the time stamp is crucial when generating the audit information to ensure accountability.
O.AUDIT_PROTECTION	The audit trail must be protected so that only authorized administrators may access it or delete it. FAU_SAR.2 ensures that only authorized administrators have read access to audit information, and FAU_STG.1 ensures that audit information is not modified and protects it from unauthorized deletions. FAU_STG.3 and FAU_STG.4 ensure that audit information is not lost because of shortage of resources.
O.AUDIT_REVIEW	FAU_SAR.1 provides the ability for an authorized administrator to efficiently review audit records. This requirement also mandates the audit information be presented in a manner that is suitable for the authorized administrators to interpret the audit trail. FAU_SAR.3 complements FAU_SAR.1 by providing the authorized administrators the flexibility to specify criteria that can be used to search or sort the audit records residing in the audit trail. FAU_SAR.3 requires the authorized administrators be able to establish the audit review criteria based on a user and identifier, date and time, so that the actions of a user can be readily identified and analysed. Allowing the

	authorized administrators to perform searches or sort the audit records based on dates, times, type of events, and success and failure of these events, provides the capability to extract the user activity to what is pertinent at that time in order to facilitate the administrator's review. It is important to note that the intent of sorting in this requirement is to allow the authorized administrators the capability to organize or group the records associated with a given criteria. FAU_STG.3 allows the authorized administrator to be alerted of the possible audit data loss if the audit trail exceeds an authorized administrator selectable, pre-defined limit.
O.CRYPTO_NET	Cryptographic services are provided in the TOE by the OpenSSL library (see sec. 8.1.6). The TOE uses the Secure Shell (SSH) V2.0 cryptographic network protocol for operating network services securely over an unsecured network. SSH V2.0 provides a secure channel over an unsecured network in a client-server architecture, connecting an SSH client application with an SSH server. Only SSH V2.0 protocol specification can be used by the TOE being the SSH V1.x version totally disabled. The following functional requirements in the area of cryptographic operations are addressed: ✓ Data encryption, decryption and cryptographic signatures for secure communication over untrusted network [FCS_COP.1(1)]; ✓ Key management services supporting the dual-factor authentication [FCS_CKM.1(2), FCS_CKM.1(3), FCS_CKM.1(4)]; ✓ Other key management services [FCS_CKM.1(1), FCS_CKM.1(3), FCS_CKM.1(4), FCS_CKM.2, FCS_CKM.4_EXT].
O.CRYPTO_MEDIA	Cryptographic services are provided in the TOE by the LibGCRYPT and LibCRYPTSETUP libraries. Specific functional requirements in the area of cryptographic

	operations to ensure secure export of information and dual-factor authentication, are addressed: ✓ Key management [FCS_CKM.1(2)]; ✓ Data encryption, decryption and cryptographic signatures [FCS_COP.1(2)].
O.TRUSTED_CHANNEL	The TSF must moreover be able to ensure a trusted Inter-TSF channel between itself and another trusted IT product [FTP_ITC.1] protecting the user data transferred from disclosure [FDP_UCT.1] and undetected modification [FDP_UIT.1]. The TSF also ensure the TOE connects to another trusted IT product accordingly to the SSH v2.0 Protocol Standard [FCS_CKM.1(1), FCS_CKM.1(3), FCS_CKM.1(4), FCS_CKM.2, FCS_CKM.4_EXT, FCS_COP.1(1)].
O.DISCRETIONARY_ACCESS	Access to TOE resources is determined by the Discretionary Access Control policy. FDP_ACC.1 enforces the Discretionary Access Control (DAC) policy on all subjects and all named objects identified and all operations among them and covered by DAC policy. FDP_ACF.1 define the Discretionary Access Control rules to determine if any operation between subjects and named objects is allowed. These rules are based on the identity of the users and their group memberships. FIA_USB.1 defines the associations between user security attributes (see FIA_ATD.1) and subjects acting on behalf of that user. The access control policy decisions are based on the associated security attributes. FMT_MSA.3 ensures that the TOE provides protection by default for all objects at creation time. This may allow authorized users to explicitly specify the desired access controls upon the object at its creation, provided that there is no window of vulnerability through which unauthorized access may be gained to newly-created objects.

O.DISCRETIONARY_USER_CONTROL	To allow authorized users to specify which resources may be accessed, the TOE must provide the ability for object security attributes to be changed and revoked. FMT_MSA.1 restricts the ability to change the value of object security attributes to authorized administrators and owners of objects. FMT_REV.1(2) restricts the ability to revoke security attributes of objects to authorized administrators and owners of these objects.
O.DISPLAY_BANNER	Before identification and authentication and the establishment of a user session, the TOE allows limited access by any potential users of the operating system in order to convey warnings and agreements for its use. Through this limited access before establishing a user session, the TSF displays an authorized, administrator-specified advisory notice and consent warning message regarding unauthorized use of the TOE [FTA_TAB.1]. In typical applications a user who continues session establishment procedures (including their successful identification and authentication) after display of the notice and warning banner effectively acknowledges the banner content and consents to the stated conditions. This banner of information can be critical in supporting legal actions related to the use of the TOE.
O.ENFORCEMENT	The TSF must be designed and implemented in a manner which ensures that the organizational policies are enforced in the target environment. The TSF must provide the authorized administrator with a self test utility [FPT_TST.1] for the TSF and TSF data integrity check.
O.MANAGE	In a variety of ways the TOE supports authorized administrators in the management of security functions, security attributes and data while also restricting unauthorized use. For example, the TOE provides for and restricts the following actions to authorized administrators only (except where specifically noted):

	• Disable and enable the audit functions, and specify which events are audited [FMT_MOF.1(1)], • Manage the values of user security attributes [FMT_MOF.1(2)], • Change the value of object security attributes. (Object owner is also allowed to perform this action.) [FMT_MSA.1], • Provide restrictive default values for security attributes, and specify alternative initial values to override the default values when an object or information is created. [FMT_MSA.3], • The management aspects of the individual SFRs (excluded FMT_MTD.1(6)) are specified with all iterations of FMT_MTD.1, • Revoke security attributes associated with the users within the TSC. [FMT_REV.1 (1)], • Revoke security attributes of objects within the TSC. (Object owner is also allowed to perform this action.) [FMT_REV.1(2)], • Specify an expiration time for authorized user authentication data. [FMT_SAE.1], • FMT_SMR.1 defines the rights management for the different management aspects, • FMT_SMF.1 provides a list of the management functions specified in this ST and is required as a dependency for the management functions.
O.PROTECT	O.PROTECT requires mechanisms be provided by the TOE to protect user data and resources. FIA_SOS.1 prescribes the maximum probability for guessing authentication data that must be satisfied. If a user can't authenticate, he or she will not have the ability to access user data and resources. This protection mechanism is also valid for LUKS devices. FIA_UAU.7 ensures that no feedback that affects the ability of users to circumvent the authentication mechanism is presented during the authentication process. The TOE is allowed to provide information

	that would allow the user to use the authentication mechanism in a correct manner (e.g., press CTRL-ALT-DELETE, slide card quickly, centre your finger and press firmly, speak louder and slowly), but not provide information that may allow alteration to their presentation that would thwart the mechanism. FMT_MSA.2 ensures that only valid configuration values are accepted for security attributes, supporting the protection of the TSF by avoiding miss-configuration. To protect user data and resources, FDP_ACC.1, FDP_ACF.1, and FMT_REV.1(2) require a Discretionary Access Control policy and rules that ensures the correct access to objects by subjects acting on behalf of users. To ensure that user data are not disclosed before a resource is reused, FDP_RIP.2 requires that the shared memory and operating system controlled files as well as resources are not available to another user thus protecting the user data. FMT_MTD.1(6) ensures that the authentication data are protected. No entity is allowed to read authentication data and the TSF must prevent any attempt to read it.
O.RESIDUAL_INFORMATION	FDP_RIP.2 ensures the contents of resources are not available upon allocation of the resource to all objects
O.USER_AUTHENTICATION	FIA_UAU.2 plays a role in satisfying this objective by ensuring that every user is authenticated before any other action will be allowed by the TSF. Multiple identification and authentication mechanisms are allowed as specified in FIA_UAU.5. FIA_UAU.6 ensures that the authorized user changing his authentication data re-authenticates before he or she is allowed to proceed. The authentication data to be changed refer to the following: • User's password; • Passphrase of the LUKS device involved in a dual-factor authentication;

	• Passphrase of the key pair involved in a dual-factor authentication. To verify the claimed identity of an authorized user, FIA_SOS.1 prescribes the metrics that must be satisfied. It provides the mechanism that will verify the secret for user authentication. In particular FIA_SOS.1 requires that: • For password-based authentication and for LUKS-formatted devices, the authentication mechanism provide the ability for authorized users to have a "secret" of at least 16 characters in length, consisting of any combination of upper and lower case letters, numbers, and punctuation; • For dual-factor authentication, the authentication mechanism provide the ability for authorized users to have a LUKS-encrypted USB token containing a DSA (or RSA) private key. Both USB token and private key are protected by different passphrases known only by the owner. The dual-factor authentication can only be activated after a successful password-based authentication [FCS_CKM.1(3), FCS_CKM.1(4), FCS_CKM.4_EXT, FCS_COP.1(1)]; • For public-key-based authentication, the authentication mechanism provide the ability for authorized users to have a DSA (or RSA) private key which is protected by a passphrase known only by the owner [FCS_CKM.1(3), FCS_CKM.1(4), FCS_CKM.4_EXT].
O.USER_IDENTIFICATION	FIA_UID.2 plays a role in satisfying this objective by ensuring that every user identify himself before any other action will be allowed by the TSF.
O.RUNTIME.PROTECTION	Using the runtime protection mechanisms offered to applications is specified by [FPT_FLS.1(1), FPT_FLS.1(2)].

Table 9: Security objectives for the TOE rationale

6.9.3 Security requirements dependencies analysis

The following table demonstrates the dependencies of SFRs modelled in CC Part 2 and how the SFRs for the TOE resolve those dependencies:

Security Functional Requirement	Dependencies	Resolution
FAU_GEN.1	FPT_STM.1	FPT_STM.1
FAU_GEN.2	FAU_GEN.1	FAU_GEN.1
	FIA_UID.1	FIA_UID.2
FAU_SAR.1	FAU_GEN.1	FAU_GEN.1
FAU_SAR.2	FAU_SAR.1	FAU_SAR.1
FAU_SAR.3	FAU_SAR.1	FAU_SAR.1
FAU_SEL.1	FAU_GEN.1	FAU_GEN.1
	FMT_MTD.1	The dependency on FMT_MTD.1 is resolved by FMT_MTD.1(1) specifying the management aspect applicable to this SFR.
FAU_STG.1	FAU_GEN.1	FAU_GEN.1
FAU_STG.3	FAU_STG.1	FAU_STG.1
FAU_STG.4	FAU_STG.1	FAU_STG.1
FCS_CKM.1(1)	[FCS_CKM.2 or FCS_COP.1]	FCS_CKM.2 FCS_COP.1(2)
FCS_CKM.1(2)	[FCS_CKM.2 or FCS_COP.1]	FCS_CKM.2 FCS_COP.1(2)
FCS_CKM.1(3)	[FCS_CKM.2 or FCS_COP.1]	FCS_CKM.2 FCS_COP.1(2)
FCS_CKM.1(4)	[FCS_CKM.2 or FCS_COP.1]	FCS_CKM.2 FCS_COP.1(2)

Security Functional Requirement	Dependencies	Resolution
FCS_CKM.2	[FDP_ITC.1 Import of user data without security attributes, or FDP_ITC.2 Import of user data with security attributes, or FCS_CKM.1 Cryptographic key generation]	FCS_CKM.1(1) FCS_CKM.1(2) FCS_CKM.1(3) FCS_CKM.1(4)
FCS_CKM.4_EXT	[FDP_ITC.1 Import of user data without security attributes, or FDP_ITC.2 Import of user data with security attributes, or FCS_CKM.1 Cryptographic key generation]	FCS_CKM.1(1) FCS_CKM.1(2)
FCS_COP.1(1)	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1]	FCS_CKM.1(1) FCS_CKM.1(3) FCS_CKM.1(4)
FCS_COP.1(2)	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1]	FCS_CKM.1(2)
FDP_ACC.1	FDP_ACF.1	FDP_ACF.1
FDP_ACF.1	FDP_ACC.1	FDP_ACC.1
	FMT_MSA.3	FMT_MSA.3
FDP_RIP.2	No dependencies.	
FDP_UCT.1	[FTP_ITC.1 Inter-TSF trusted channel, or FTP_TRP .1 Trusted path] [FDP_ACC.1 Subset access control, or FDP_IFC.1 Subset information flow control]	FTP_ITC.1 FDP_ACC.1
FDP_UIT.1	[FDP_ACC.1 Subset access control, or FDP_IFC.1 Subset information flow control] [FTP_ITC.1 Inter-TSF trusted channel, or FTP_TRP .1 Trusted path]	FTP_ITC.1 FDP_ACC.1

Security Functional Requirement	Dependencies	Resolution
FIA_AFL.1	FIA_UAU.1	FIA_UAU.2
FIA_ATD.1	No dependencies.	
FIA_SOS.1	No dependencies.	
FIA_UAU.2	FIA_UID.1	FIA_UID.2
FIA_UAU.5	No dependencies.	
FIA_UAU.6	No dependencies.	
FIA_UAU.7	FIA_UAU.1	FIA_UAU.2
FIA_UID.2	No dependencies.	
FIA_USB.1	FIA_ATD.1	FIA_ATD.1
FMT_MOF.1(1)	FMT_SMR.1	FMT_SMR.1
	FMT_SMF.1	FMT_SMF.1
FMT_MOF.1(2)	FMT_SMR.1	FMT_SMR.1
	FMT_SMF.1	FMT_SMF.1
FMT_MSA.1	[FDP_ACC.1 or FDP_IFC.1]	FDP_ACC.1
	FMT_SMR.1	FMT_SMR.1
	FMT_SMF.1	FMT_SMF.1
FMT_MSA.2	FDP_ACC.1	FDP_ACC.1
	FMT_MSA.1	FMT_MSA.1
	FMT_SMR.1	FMT_SMR.1
FMT_MSA.3	FMT_MSA.1	FMT_MSA.1
	FMT_SMR.1	FMT_SMR.1
FMT_MTD.1(1)	FMT_SMR.1	FMT_SMR.1

Security Functional Requirement	Dependencies	Resolution
	FMT_SMF.1	FMT_SMF.1
FMT_MTD.1(2)	FMT_SMR.1	FMT_SMR.1
	FMT_SMF.1	FMT_SMF.1
FMT_MTD.1(3)	FMT_SMR.1	FMT_SMR.1
	FMT_SMF.1	FMT_SMF.1
FMT_MTD.1(4)	FMT_SMR.1	FMT_SMR.1
	FMT_SMF.1	FMT_SMF.1
FMT_MTD.1(5)	FMT_SMR.1	FMT_SMR.1
	FMT_SMF.1	FMT_SMF.1
FMT_MTD.1(6)	FMT_SMR.1	FMT_SMR.1
	FMT_SMF.1	FMT_SMF.1
FMT_MTD.1(7)	FMT_SMR.1	FMT_SMR.1
	FMT_SMF.1	FMT_SMF.1
FMT_REV.1(1)	FMT_SMR.1	FMT_SMR.1
FMT_REV.1(2)	FMT_SMR.1	FMT_SMR.1
FMT_SAE.1	FMT_SMR.1	FMT_SMR.1
	FPT_STM.1	FPT_STM.1
FMT_SMF.1	No dependencies.	
FMT_SMR.1	FIA_UID.1	FIA_UID.2
FPT_STM.1	No dependencies.	
FPT_TST.1	No dependencies.	
FPT_FLS.1(1)	No dependencies.	

Security Functional Requirement	Dependencies	Resolution
FPT_FLS.1(2)	No dependencies.	
FTA_SSL.2	FIA_UAU.1	FIA_UAU.2
FTA_TAB.1	No dependencies.	
FTA_TAH.1	No dependencies.	
FTP_ITC.1	No dependencies.	

Table 10: TOE SFR dependency analysis

7 Security Assurance Requirements

The security assurance requirements for the TOE are the Evaluation Assurance Level 4 components augmented by ALC_FLR.1, as specified in [CC] part 3.

7.1 Security Assurance Requirements Rationale

The basis for the justification of EAL4 augmented with ALC_FLR.1 is the threat environment experienced by the typical consumers of the TOE. This matches the package description for EAL4 (enhanced-basic).

8 TOE Summary Specification

8.1 TOE Security Functionalities

The following sections provide details on the implementation of the security functionalities provided by the TOE. The different TOE security functions cover the various SFR classes. The primary security features of the TOE are:

- Identification and Authentication
- Auditing
- Discretionary Access Control
- Object Reuse
- Security Management
- Cryptographic services
- TSF Protection.

8.1.1 Identification and Authentication (IA)

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* command. These all rely on explicit authentication information provided interactively by an user.

- For password-based authentication, an administrative database is used. It is managed by authorized administrators, but non-administrative users are allowed to modify their own password using the *passwd* command.
- For dual-factor authentication, special configuration files are used. They are managed by authorized administrators, but users are allowed to modify passphrases for their assigned USB token (i.e. the LUKS device) and for their DSA (or RSA) key pair. The dual-factor authentication can only be activated after a successful password-based authentication on a local terminal. Even if the user knows passphrases for the USB token and for key pair, he or she is not able to access the device: then no read or write or execute operations are possible on the device.
- Password-based authentication or public-key-based authentication is possible for SSH sessions.

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.

8.1.1.1 Identification and Authentication mechanisms (IA.1)

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:

- provide login control and establish identification IDs for a subject (e.g., UID, GID, etc.),

- ensure the quality of passwords,
- consult a user's password "history" and not allowing them to re-use olds passwords,
- enforce limits for accounts,
- restrict the use of the root account to certain terminals,
- restrict the use of the *su* command,
- enforcement of locking of accounts after failed login attempts
- manage the dual-factor authentication by accessing the USB token (i.e. LUKS device) and performing the key signature verification.

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. After a successful password-based authentication on a local terminal, the user can be forced to go trough dual-factor authentication. Then, it is prompted to insert its assigned USB token and to type in the passphrases for the USB token and for the DSA (or RSA) key pair.

To enable a dual-factor authentication for an user, the authorized administrator must first create a LUKS-formatted USB token and then has to generate an RSA (or DSA) key pair. The private part of the key pair is stored on the USB token. The public part is copied in the local machine: it is owned by root user only and it is protected by DAC. To protect the USB token against unauthorized accesses, its header section is stored in the local machine and protected by DAC (owned by root user only).

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.

This security function covers the SFRs FMT_SMR.1, FIA_SOS.1, FIA_UID.2, FIA_UAU.2, FIA_UAU.5, FIA_UAU.6, FAU_GEN.1, FIA_AFL.1, FIA_USB.1, FIA_UAU.7, FTA_TAB.1 and FTA_TAH.1.

8.1.1.1.1 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 DSA) 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 "publickey" authentication using the private key on the client side and the public key found on the server side. The operations performed during the publickey authentication is defined in RFC 4252 chapter 7.

Users 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.

This security function covers the SFRs of FIA_UAU.2, FIA_UAU.5, FIA_UID.2, FIA_SOS.1.

8.1.1.2 User Identity and Role Changing (IA.2)

Users can change their identity (i.e., switch to another identity) using the *su* command. 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 appropriately 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 common authentication mechanism to validate the supplied authentication. After a successful password-based authentication on a local terminal, the user can be forced to go trough dual-factor authentication. Then, it is prompted to insert its assigned USB token and to type in the passphrases for the USB token and for the DSA (or RSA) key pair.

This security function covers the SFR FIA_USB.1.

8.1.1.3 User Session Lock (IA.3)

Sessions can be locked by users voluntarily via the *vlock* application. To unlock the locked session, the user must type in his or her password; user'a authentication is then performed via PAM.

This security function covers the SFR FTA_SSL.2.

8.1.1.4 Management of Authentication Data (IA.4)

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.

Each TOE instance within the distributed deployed system maintains its own administrative database by making all administrative changes on the local TOE instance. TOE administration ensures that all machines within the distributed deployed configuration are configured in accordance with the requirements defined in this Security Target.

The file /etc/passwd contains for each user the user's name, the id of the user, an indicator whether the password of the user is valid, the principal group id of the user and other (not security relevant) information. The file /etc/shadow contains for each user a hash of the user's password, the userid, the time the password was last changed, the expiration time as well as the validity period of the password

and some other information that are not subject to the security functions as defined in this Security Target.

Authorized users are allowed to change their passwords by using the *passwd* command. This application is able to read and modify the contents of /etc/shadow only for the user's password entry, which would ordinarily be inaccessible to a non-privileged user process (this implies that the TSF does not rely on the strength of the hashing algorithm to protect the passwords). 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.

The configuration of this functional aspect is done by modifying the parameters for the PAM modules in the PAM configuration files stored in /etc/pam.d/. By default, the number of consecutive unsuccessful authentication attempts before the account will be suspended (for authorized administrators) or disabled (for authorized users) are set to 10 for authorized administrator and to 5 for other users.

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

When the dual-factor authentication is enabled for a user, the following authentication data are also involved:

- Passphrase of the LUKS-formatted USB token – This passphrase is stored in the header section of the LUKS device [**LUKS**] and the user can be forced to updated it at first use of the USB token, or such passphrase can be forced to sync with the user's password;
- Passphrase of the RSA (or DSA) key pair – This passphrase is stored in the keys themselves [**FIPS186-2, FIPS186-4**] and the user can be forced to updated it at first use of the keys.
  - The OpenSSL package [**OMSecPol**] provides cryptographic primitives for managing the sign and verification process involving the RSA (or DSA) key pair. These cryptographic primitives are invoked in the PAM stack
  - The OpenSSL package also provides the *openssl* application for generating the RSA (or DSA) key pair to be used for the dual-factor authentication.

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 can be specified also for remote logins (e.g., via SSH).

This security function covers the SFRs FTA_TAH.1, FMT_SMR.1, FTA_TAB.1, FIA_UAU.2, FIA_SOS.1, FIA_UID.2, FAU_GEN.1, FIA_AFL.1, FIA_ATD.1, FIA_USB.1, FIA_UAU.7.

8.1.2 Audit Generation and Review (AU)

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

8.1.2.1 Audit Generation and Processing (AU.1)

The kernel interface which 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, is halted or the audit daemon executes an administrator-specified notification action depending on the configuration of the audit daemon.

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

The authorized administrator can define the events to be audited 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.

This security function covers the SFRs FAU_GEN.1, FAU_GEN2, FAU_SEL.1, FAU_SAR.2, FAU_STG.1, FAU_STG.3, FAU_STG.4 and FMT_MTD.1 (for audit data).

8.1.2.2 Audit Review (AU.2)

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 *root* only.

This security function covers the SFRs FAU_GEN.1, FAU_GEN.2, FAU_SEL.1, FAU_SAR.1, FAU_SAR.3.

8.1.3 Discretionary Access Control (DA)

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.

8.1.3.1 General DAC Policy (DA.1)

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 which 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 which 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.

This security function covers the SFRs FDP_RIP.2, FMT_REV.1(1), FMT_REV.1(2), FMT_MSA.1, FMT_MSA.2, FMT_MSA.3.

8.1.3.2 Permission bits (DA.2)

The TOE supports standard UNIX permission bits to provide one form of DAC for file system objects in all supported file systems. There are three sets of three bits that define access for three categories of users: the owning user, users in the owning group, and other users. The three bits in each set indicate the access permissions granted to each user category: one bit for read (r), one for write(w) and one for execute (x). Note that write access to file systems mounted as read only (e.g. CD-ROM) is always rejected. Also, write access to file system objects marked as immutable is always rejected. The SAVETXT attribute is used for world-writeable temp directories preventing the removal of files by users other than the owner.

Each process has an inheritable “umask” attribute which is used to determine the default access permissions for new objects. It is a bit mask of the user/group/other read/write/execute bits, and specifies the access bits to be removed from new objects. For example, setting the umask to “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.

This security function covers the SFRs FAU_SAR.2, FDP_ACC.1, FIA_USB.1 and FDP_ACF.1.

8.1.3.3 Access Control Lists (ACLs) (DA.3)

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:

- ext3 and ext4;
- 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 inherits the default ACL from its parent directory. When an object is created within a directory and the ACL is not defined with the function creating the object, the new object inherits the default ACL of its parent directory as its initial ACL.

This security function covers the SFRs FDP_ACC.1, FDP_ACF.1, FMT_MSA.1, FMT_SMF.1, FMT_MSA.3 and FIA_USB.1.

8.1.3.4 IPC objects (DA.4)

The TOE implements the following standard types of IPC mechanisms:

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

Access to the above mentioned IPC mechanisms are governed by UNIX permission bits. 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.

This security function covers the SFRs FDP_ACC.1, FDP_ACF.1.

8.1.4 Object Reuse Functionality (OR.1)

Object Reuse is the mechanism that protects against scavenging, or being able to read information that is left over from a previous subject's actions. Explicit initialization is appropriate for most TSF-managed abstractions, where the resource is implemented by some TSF internal data structure whose contents are not visible outside the TSF: queues, datagrams, pipes, and devices. These resources are completely initialized when created, and have no information contents remaining. Explicit clearing is used in FIN.X-RTOS only for directory entries, because they are accessible in two ways: through TSF interfaces both for managing directories and for reading files. Because this exposes the internal structure of the resource, it must be explicitly cleared on release to prevent the internal state from remaining visible. Storage management is used in conjunction with explicit initialization for object reuse on files, and processes. This technique keeps track of how storage object such as File System, IPC or Memory object is used, and whether it can safely be made available to a subject.

This security function covers the SFR FDP_RIP.2.

8.1.5 Security Management (SM.1)

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 is 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.

The TOE provides security management applications for all security-relevant settings listed throughout this ST, i.e. all FMT_MSA.1 and FMT_MTD.1 iterations.

This security function covers the SFRs FMT_SMR.1, FMT_MSA.1, FMT_MSA.2, FMT_MSA.3, FMT_SAE.1, FMT_SMF.1, FPT_STM.1, FIA_SOS.1, FIA_UAU.7, FDP_ACC.1, FDP_ACF.1, all iterations of FMT_MOF.1, FMT_MTD.1 and FMT_REV.1.

8.1.6 Cryptographic services (CS)

8.1.6.1 Secured network communication channels (CS.1)

The TOE provides cryptographically secured network communication channels to allow remote users to interact with the TOE:

- The OpenSSH package provides applications allowing the TOE to establish secure remote connections. OpenSSH implements the SSH v2.0 protocol. The OpenSSL package [**OMSecPol**] provides to the OpenSSH package the cryptographic primitives for implementing the above mentioned cryptographic communication protocol;
- *ssh* and *sshd* are the client-server pair provided by OpenSSH that allow users to login from remote IP systems using secure encrypted channels. *sshd* is a daemon configured to work through the PAM framework. Users may employ such utilities for interactive sessions as well as for non-interactive sessions; the console provided by *ssh* provides the same environment as a local console;
- The *ssh-keygen* application provided by OpenSSH generates, manages, and converts authentication keys for SSH sessions. These keys are used only for public-key-based authentication method. The key generation mechanism uses the Linux kernel random number generator. The evaluated configuration permits the import of externally-generated RSA (and DSA) key pair.

The cryptographic primitives used by OpenSSH aplications for implementing the above mentioned cryptographic communication protocols – also including the random number generator seeded by /dev/random or /dev/urandom to generate cryptographic keys – are provided by the OpenSSL library [**OMSecPol**].

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

- Establishing a secure communication channel using the following cryptographic functions provided by the SSH v2.0 protocol:
  - Encryption as defined in section 6.3 of RFC 4253 - the keys are generated using the Linux kernel random number generator;
    - Symmetric keys are used to encrypt the entire connection (Public/private asymmetrical key pairs are only used for authentication; see next);
    - A symmetric key is session-based and constitutes the actual encryption for the data sent between server and client. Once a session is established, data are encrypted with this symmetric key;
    - Simmetric keys are zeroized when no longer required. Zeroization of these keys is performed by means of the the OpenSSL APIs [**OMSecPol**], specifically the *RSA_free()* and *DSA_free()* functions which in turn invoke the *OPENSSL_cleanse()* function for filling the key's memory space with a string of 0's and then invoke the *OPENSSL_free()* function for freeing that memory space.
  - Diffie-Hellman key exchange as defined in section 6.1 of RFC 4253;
    - Both client (*ssh*) and server (*sshd*) contribute toward establishing a symmetric key which is created through a process known as a key exchange algorithm. This exchange results in the server and client both arriving at the same key independently by sharing certain pieces of public data and manipulating them with certain secret data.
  - The keyed hash function for integrity protection as defined in section 6.4 of RFC 4253 and its updating RFC 6668.
- Performing user authentication using the standard password-based authentication method the TOE provides for users (password authentication method as defined in chapter 5 of RFC 4252);
- Performing user authentication using a RSA (or DSA) public-key-based authentication method (public-key-based authentication method as defined in chapter 7 of RFC 4252)
  - Public keys are derived by the *ssh-keygen* application starting from the respective private keys;
  - Private keys [**FIPS186-2, FIPS186-4**] may be generated by the *ssh-keygen* application by means of the OpenSSL APIs, or they may be generated by the *openssl* application.
- Checking the integrity of the messages exchanged.

This security function covers the SFRs of: FCS_CKM.1(1), FCS_CKM.1(3), FCS_CKM.1(4), FIA_SOS.1, FCS_CKM.2, FCS_CKM.4_EXT, FCS_COP.1(1), FTP_ITC.1, FDP_UCT.1, FIA_UAU.5, and FDP_UIT.1.

8.1.6.2 Confidentiality protected data storage (CS.2)

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.

Passphrases for LUKS-formatted devices must satisfy specific complexity criteria.

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 user-level application that allow 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 and inject it into the kernel;
3. Set up the mapping between the block device and the volume key.

Using the *cryptsetup* application, multiple volume keys can be managed for the samed encrypted device [**LUKS**]. Zeroization of a volume key is performed by *cryptsetup* as follow:

- ✓ A portion of the volume key section is overwritten using the Gutmann method (http://www.cs.auckland.ac.nz/~pgut001/pubs/secure_del.html) while the remaining part is overwritten with random data extracted from */dev/random* or */dev/urandom* sources;
- ✓ At the end of previous step, the overall volume key section is again overwritten with random data extracted from */dev/random* or */dev/urandom* sources;
- ✓ Finally, the volume key's memory space is filled with a string of 0's.

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.

This security function covers the SFRs of: FCS_CKM.1(2), FCS_CKM.4_EXT, FCS_COP.1(2), FIA_SOS.1, FDP_UCT.1, FIA_UAU.5, and FDP_UIT.1.

8.1.7 TSF protection (TP)

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 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.

The boot image for each host with the evaluated TOE is adequately protected using proper DAC permission settings. In addition, checksum of the boot image is required to match so that the inability to retrieve a pre-calculated hash shall force the application to terminate immediately.

8.1.7.1 TSF Invocation Guarantees (TP.1)

All TOE protected resources are managed by the TSF. Because all TSF data structures are protected, these resources can be directly manipulated only by the TSF, through defined TSF interfaces. This satisfies the condition that the TSF must be "always invoked" to manipulate protected resources.

Resources managed by the kernel software can only be manipulated while running in kernel mode. Processes run in user mode and can call functions (i.e. system calls) of the kernel only by means of the sysenter assembly language instruction (System call parameters are passed directly in registers *ebx*, *ecx*, *edx*, *esi*, and *edi*. The *eax* register contains the system call number) The hardware and the kernel software handle these events and ensure that the kernel is entered only at pre-determined locations, and within pre-determined parameters. All kernel-managed resources are protected such that only the kernel software is able to manipulate them.

Trusted processes implement resources managed outside the kernel. The trusted processes and the data defining the resources are protected as described above depending on the type of interface. For directly invoked trusted processes the program invocation mechanism ensures that the trusted process always starts in a protected environment at a predetermined point. Other trusted process interfaces are started during TOE initialization and use well defined protocol or file system mechanisms to receive requests.

Some system calls or parameter of system calls are reserved for trusted processes. When called the kernel checks that the calling process runs with an effective userid of 0.

This function contributes to satisfy the security requirement ADV_ARC.1.

8.1.7.2 Kernel (TP.2)

The TOE software consists of a privileged kernel and a variety of non-kernel components (trusted processes).

The kernel operates on behalf of all processes (subjects). The kernel runs in the CPU's privileged mode and has access to all system memory. All kernel software, including kernel extensions and kernel processes, execute with kernel privileges but only defined subsystems within the kernel are part of the TSF. The kernel is entered by some event that causes a context switch such as a system call, I/O interrupt, or a program exception condition.

Upon entry the kernel determines the function to be performed, performs it, and, when finished, performs another context switch to return to user processing (eventually on behalf of a different subject) (TP1.1).

The kernel is shared by all processes, and manages system wide shared resources. It presents the primary programming interface for the TOE in the form of system calls.

This function contributes to satisfy the security requirement ADV_ARC.1.

8.1.7.3 Kernel Modules (TP.3)

The TOE supports dynamically loadable kernel modules that are loaded automatically on demand. Kernel modules are actually a part of the kernel that is not resident but loaded as part of the kernel when needed.

Whenever a program wants the kernel to use a feature that is only available as a loadable module, and if the kernel hasn't got the module installed yet, the kernel will take care of the situation and make the best of it.

This function contributes to satisfy the security requirement ADV_ARC.1.

8.1.7.4 Trusted Processes (TP.4)

Trusted processes in the TOE are processes running in user mode but with root privileges.

A trusted process is distinguished from other user processes by the ability to affect the security policy. Some trusted processes implement security policies directly (e.g., identification and authentication) but many are trusted simply because they operate in an environment that confers the ability to access TSF data (e.g., programs run by authorized administrators or during TOE initialization).

Any program executed with root privileges has the ability to perform the actions of a trusted process. It is therefore important that a site operating the TOE strictly controls those programs and prohibits that those programs are modified or that programs from untrusted sources are executed with root privileges.

Trusted processes are not part of the kernel and (except for those processes that perform TOE initialization and identification and authentication) are not part of the TSF itself.

Trusted processes provide a contribution to security management and identification and authentication. For identification and authentication they contribute to satisfy the security functional requirements FIA_UAU.2, FIA_UAU.7 and FIA_UID.2.

This function also contributes to ADV_ARC.1.

8.1.7.5 TSF Databases (TP.5)

TOE uses configuration files, in the following identified as "databases", to control user permissions, system applications, daemons, services, and other administrative tasks. By means of these databases the TOE accesses the list of users and groups in the system, and manage file permissions (that is, determine if a file can be opened by a specific user, according to the permissions). With the exception of those files that a user can read, but not write, all of these databases shall only be accessible to authorized administrators. None of these databases shall be modifiable by an user other than an authorized administrator.

Those databases are part of the file system and therefore the file system protection mechanisms of the TOE have to be used to protect those databases from 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.

Each host system within the TOE maintains its own TSF databases. Synchronizing those databases is not performed in the evaluated configuration. If such synchronization is required by an organization it is the responsibility of an authorized administrator of the TOE.

The databases set is not a TSF, but it is part of the management of the TSF. As such it contributes to TOE management security functional requirements FMT_MSA.3 and FMT_MTD.1 (user security attributes and authentication data) as well as FMT_SMF.1.

8.1.7.6 Internal TOE Protection Mechanisms (TP.6)

All kernel software has access to all of memory, and the ability to execute all instructions. In general, however, only memory containing kernel data structures is manipulated by kernel software. Parameters are copied to and from process storage (i.e., that accessible outside the kernel) by explicit internal mechanisms, and those interfaces only refer to storage belonging to the process that invoked the kernel (e.g., by a system call).

Functions implemented in trusted processes are more strongly isolated than the kernel. Because there is no explicit sharing of data, as there is in the kernel address space, all communications and interactions between trusted processes take place explicitly through files and similar mechanisms.

This encourages an architecture in which specific TSF functions are implemented by well-defined groups of processes.

This function contributes to satisfy the security requirement ADV_ARC.1.

8.1.7.7 Testing the TSF Mechanisms (TP.7)

The TOE provides a tool for the authorized administrator that allows him to run a system self test to demonstrate correct operation of the TSF. This tool performs tests on:

- the integrity of TSF data,
- the integrity of stored TSF executable code,
- correct operation of the cryptographic algorithms.

- correct operation of the DAC mechanism.

The tool generates a report on the tests performed and the results that those test had. The report is generated in human readable format and may be stored in a file or directed to a printer. This function contributes to satisfy the security requirement FPT_TST.1.

8.1.7.8 Protection Mechanisms against buffer overrun vulnerability

The TOE provides functionality to mitigate the effects of a potentially present buffer overrun vulnerability in applications. This mitigation mechanism covers the following aspects:

- Configuring the memory holding the stack to be readable and writable only, but not executable. By default, all processes and threads use stacks which are non-executable.
- Performing address space randomization which affects position independent code (PIC) as well as position independent executables (PIE). All shared libraries are necessarily PIC, whereas only dedicated TSF applications are PIE. Users may compile their applications as PIE to allow address space randomization protect their applications.
- Marking the runtime memory of all parts of a binary as read-only apart from heap data before the loaded application gains control. This support is enabled for dedicated TSF applications and specifically compiled user applications. Partial protection is also possible which implies that also the .got.plt section is marked read/writable.

This security function covers the SFR of FPT_FLS.1(1), FPT_FLS.1(2).

9 Abbreviations and References

9.1 ABBREVIATIONS

ACL	Access Control Lists
AES	Advanced Encryption Standard
ANSI	American National Standards Institute
API	Application Program Interface
ASCII	American Standard Code for Information Interchange
CBC	Cipher Block Chaining
CC	Common Criteria
CD	Compact Disk
CD-ROM	Compact Disk - Read Only Memory
CPU	Central Processing Unit
CSCI	Computer Software Configuration Item
CTR	Cipher Counter mode
DAC	Discretionary Access Control
DSA	Digital Signature Algorithm
DVD	Digital Versatile Disk
EAL4	Evaluation Assurance Level 4
ESSIV	Encrypted Salt-Sector Initialization Vector
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
IPC	Inter-Process Communication
ISO	International Organization for Standardization

IT	Information & Technology
LAF	Lightweight Audit Framework
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
PBKDF2	Password-Based Key Derivation Function 2
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
S&F	Success & Failure
SE	Security Enhanced
SFP	Security Function Policy
SFR	Security Functional Requirement
SHA	Secure Hash Algorithm
SHA1	Secure Hashing Algorithm 1
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

9.2 REFERENCES

[CC] Common Criteria for Information Technology Security Evaluation, CCMB-2012-09-001 to CCIMB-2012-09-003, Version 3.1 Revision 4, September 2012 (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 .

[FIPS186-2] FIPS 186-2: Federal Information Processing Standards Publication – Digital Signature Standard (DSS) – October, 2001, http://csrc.nist.gov/publications/fips/fips186-2/fips_186-2.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.

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

[GPOSPP] US Government Protection Profile for General-Purpose Operating Systems in a Networked environment. Version 1.0 as of 2010-08-30.

[OSPP] Operating system Protection Profile Version 2.0.

[OMSecPol] OpenSSL FIPS 140-2 Security Policy Version 2.0.9.

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

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