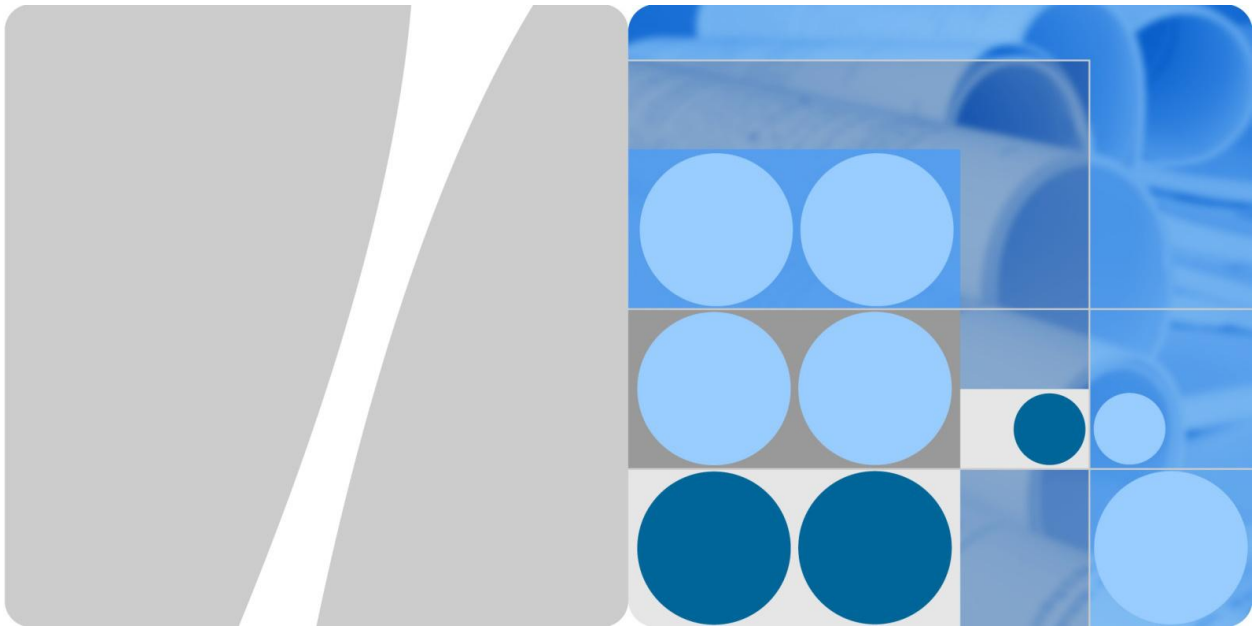

Huawei EulerOS Version 2.0 Common Criteria Evaluation

Security Target

(Against BSI PP)

Version	0.13
Status	Released
Last update	2019-04-13
Classification	Public

HUAWEI TECHNOLOGIES CO., LTD.

Trademarks and Permissions

 and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Huawei Technologies Co., Ltd.

Address: Huawei Industrial Base
Bantian, Longgang
Shenzhen 518129
People's Republic of China

Website: http://e.huawei.com

About This Document

Purpose

This document provides description about ST (Security Target) against BSI PP.

Change History

Changes between document issues are cumulative. The latest document issue contains all the changes made in earlier issues.

Date	Revision Version	Section Number	Change Description	Author
2017.06.06	0.1	All	Initial Draft	
2017.08.02	0.2	All	Add extended package (AM), FCS_COP.1(DP,IV), RNG(SSL-DFLT, DM-INIT, DM-RUN, NSS, IV); Supported algorithms sorted; Change ‘trusted boot’ to ‘secure boot’;	
2017.09.29	0.3	All	Changed based on internal review results; SSSD removed; Assignment/selection added;	
2018.01.10	0.4	All	‘A.PROTECT.INTEGRITY’ replaced with ‘A.PROTECT.NMENV’ according to [OSPP-TB]; security features in logical boundaries re-ordered; section “Runtime Protection Mechanisms” removed from TSS and logical boundaries ; Format of Table 18 changed;	

			'O.CRYPTO.USERDATA' renamed to 'O.CP.USERDATA', and 'FCS_COP.1(DP)' renamed to 'FCS_COP.1(CP)';	
2018.03.23	0.5	FMT_MTD.1(IV-ACT)	No user can query or change the action on integrity violation, and an application note is added;	
		"Non-TOE Hardware, Software, Firmware supported"	More supported server models are added.	
		All	Review info removed.	
2018.04.20	0.6	FCS_CKM.1(DSA)	DSA usage in *OpenSSH* is moved outside of the scope of this ST;	
		All	Typographical convention is added in section "Security requirements"; Changes according to EE's feedback.	
2018.05.21	0.7	FMT_MTD.1(AE)	Format some text as AMR suggested;	
		FMT_MTD.1(AM-MR)	"**selection: delete, clear**" is changed to "**selection: delete**" as AMR suggested.	
		FMT_MSA.1(TSO)	CAP_FOWNER is changed to CAP_SYS_ADMIN	
		FIA_ATD.1(HU)	The example about soft token is removed from application note.	
		MT_MTD.1(NI)	CAP_NET_RAW is added for *iptables*.	
		FMT_MTD.1(IAU)	'general information' is used instead of 'authentication data', and an explanation is also added in the application note.	
2018.07.23	0.8	FMT_MTD.1(AE)	The ability to query/modify audit rules is given to only ROOT user. (uid is checked in source code)	
		"Security Policy Model" in chapter	'named objects' are classified into two types: user mode objects, kernel	

		‘Security Target Introduction’	mode objects;	
		FDP_ACC.1(PSO) Subset access control	An application note is added, stating WRITE not supported for some file systems (e.g. ISO9660).	
2018.08.21	0.9	Section ‘Non-TOE Hardware, Software, Firmware supported’	“(with TPM chip embedded)” removed.	
		FCS_CKM.1(DSA)	‘L=2048, N=224’ removed.	
		FAU_STG.4	Option ‘notify the root user’ removed.	
		FMT_MSA.3(TSO)	Add NOTE in the ‘application note’: umask does not impact System V IPCs.	
		FDP_IFF.1.3	‘Statistical analysis matching’ removed;	
		FDP_ACC.1(PSO)and FDP_ACF.1(PSO)	Application note added for vfat, as it is persistent file system but not POSIX compliant.	
2018.09.12	0.10	FAU_STG.4.1 7.1.3	Option ‘a)Stop all processes that attempt to generate an audit record;’ is removed. Description of ‘Audit log overflow protection’ is changed accordingly.	
2018.12.03	0.11	FCS_COP.1(NET)	TLS algorithms recovered based on the test result	
2019.01.02	0.12	Non-TOE Hardware, Software, Firmware supported	New Taishan models are added.	
2019.04.13	0.13	Section 1.3.2.1; Section 2;	Modified according to feedback from CB: Section 1.3.2.1 (for general hardware platform) removed; PP/ExtPackage URL added at the end of Section 2;	

Contents

Index of Tables

1 Security Target Introduction

1.1 Security Target Reference

Name	EulerOS 2.0 Security Target
Version	0.13
Date	2019-04-13
Sponsor	Huawei Technologies Co., Ltd
Developer	Huawei Technologies Co., Ltd

1.2 TOE Reference

Name	EulerOS
Version	V200R002C20 (Commercial Version Code: 2.0)

1.3 TOE Overview

1.3.1 Configurations defined with this ST

This security target documents the security characteristics of the Huawei EulerOS product (abbreviated with EulerOS throughout this document).

1.3.2 TOE Type

Huawei EulerOS is a highly-configurable Linux-based general-purpose operating system, which has been developed to provide a good level of security as required in commercial environments.

1.3.3 Non-TOE Hardware, Software, Firmware supported

Non-TOE Hardware Identification: The following physical and virtual hardware platforms, corresponding firmware, and components are supported by the TOE:

- FusionCube 2000, 6000, 6000C, 9000
- FusionServer E9000 Blade Server
- FusionServer G2500, G5500 Heterogeneous Servers
- FusionServer RH1288 V3 Rack Server
- FusionServer RH1288H V5 Rack Server
- FusionServer RH2288 V3 Rack Server
- FusionServer RH2288H V3 / V5 Rack
- FusionServer RH2488 V5 Rack Server
- FusionServer RH2488H V5 Rack Server
- FusionServer RH5288 V3 Rack Server
- FusionServer RH5885 V3 Rack Server
- FusionServer RH5885H V3 Rack Server
- FusionServer RH8100 V3 / V5 Rack Server
- FusionServer X6000, X6000 V5 High-Density Servers
- FusionServer X6800 Data Center Server
- FusionServer XH628 V3 Server Node
- Kunlun 9008, 9016, 9032
- Huawei Taishan 2280, 5280, X6000

- Linux QEMU-KVM-1.5.3 virtual platform (with virtualized TPM chip)

Note: the TPM chip used in evaluation is *Infineon SLB9665*.

1.3.4 Intended Method of Use

1.3.4.1 General-purpose computing environment

The TOE is a Linux-based multi-user multi-tasking operating system. It may provide services to several users, local or remote, at the same time. After successful login, the users gets access to a general computing environment, allowing launching user applications, issuing user commands at shell level, creating and accessing files. The TOE provides adequate mechanisms to separate the users and protect their data. Privileged commands are restricted to only administrative users.

The TOE is intended to operate in a networked environment with other instantiations of the TOE as well as other well-behaved peer systems.

It is assumed that responsibility for the safeguarding of the user data protected by the TOE can be delegated to human users of the TOE if such users are allowed to log on and spawn processes on their behalf. All user data is under the control of the TOE. The user data is stored in named objects, and the TOE can associate a description of the access rights to that object with each named object.

The TOE enforces controls such that access to data objects can only take place in accordance with the access restrictions placed on that object by its owner, and by administrative users. Ownership of named objects may be transferred under the control of the access control policies implemented by the TOE.

The TOE enforces discretionary access control policy, in which, access rights (e.g. read, write, execute) can be assigned to data objects with respect to subjects identified with their UID, GID and supplementary GIDs. Once a subject is granted access to an object, the content of that object may be used freely by the subject to influence other objects accessible to the same subject.

1.3.4.2 Operating Environment

The TOE permits one or more processors and attached peripheral and storage devices to be used by multiple applications assigned to different UIDs to perform a variety of functions requiring controlled shared access to the data stored on the system. With different UIDs, proper access restrictions to resources assigned to processes can be enforced using the access control mechanism provided by the TOE. Such usage scenarios are typical for systems accessed by processes, or by local users of the computer system, or by other users with protected access to the system.

1.3.5 Major Security Features

EulerOS is a general purpose, multi-user, multi-tasking Linux based operating system. It provides a platform for a variety of applications, including services on cloud environment.

EulerOS provides the following key security features:

- Security Audit: The TOE is able to intercept all system calls and recording the events occurred in the system. The security audit functionality also allows to configure the events to be audited, review and search the audit log retrieved.

- Cryptographic support: The TOE provides cryptographically secured communication to allow remote entities to log into the TOE. It is achieved by using the SSHv2 protocol. The TOE also provided IPSec and TLS protocols in order to secure the communications with other IT entities. Moreover, the TOE offers the possibility of encrypt stored data.

- Identification and Authentication: The TOE includes several ways to identify and authenticate the users (via the local console using username and password or via the SSH using password and public-key based authentication. The TOE also offers a password quality enforcement mechanism as well as it is able to handle failed authentication attempts.

- User Data Protection: The TOE offers a Discretionary Access Control (DAC) which allow owner of named objects to control the access permissions to these objects. Moreover, the TOE kernel implements the *IPTables* mechanism in order to provide a packet filter at network and transfer layer. Using these two mechanism the TOE offers an access control policy as well as an information flow control policy.

- Security Management: The TOE offers to the users and/or authorized administrators the possibility of modifying the configuration of TSF. The TOE allows local and remote management using by using OpenSSH.

- Protection of the TSF: The TOE has a boot and system integrity verification mechanisms which assure that any attempt to compromise the integrity of the TOE is detected. This is achieved by signed the kernel image together with its modules. Moreover, sensitive files stored in the user space are protected using *IMA appraisal.*

- TOE Access: The TOE is able to end user sessions after an inactivity period of time. This can be initiated by the TSF itself or by user request.

- Trusted Channel: Using the cryptographic communication protocols above mentioned (SSH, IPSec and TLS) the TOE is able to establish secure and trusted communication channel with other IT entities.

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

1.4 TOE Description

1.4.1 Introduction

EulerOS is a general purpose, multi-user, multi-tasking Linux based operating system. It provides a platform for a variety of applications, including services on cloud environment.

EulerOS evaluation covers a potentially distributed network of systems running the evaluated version and its configurations as well as other peer systems operating within the same management domain. The hardware platforms selected for the evaluation consist of machines that are available on market 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 EulerOS that run in kernel mode plus some trusted processes running in user mode. These are the functions that enforce the security policy as defined in this Security Target. Tools and commands executed in user mode that are used by an administrative user need also to be trusted to manage the system in a secure way, but they are not considered to be part of this TSF, just as with other operating system evaluations.

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

The TOE includes standard networking applications, such as *sshd*(8), which allow to access the TOE via cryptographically protected communication channel.

System administration tools include the standard command line tools. A graphical user interface for system administration or any other operation is not included in the evaluated configuration.

The TOE environment also includes applications that are not evaluated, but are used as unprivileged tools to access public system services. 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. Additional documentation is available that provides guidance how to set up such applications on the TOE in a secure way.

1.4.2 TOE boundaries

1.4.2.1 Physical boundaries

The TOE (version V200R002C20, a.k.a 2.0) is supplied in the form of ISO images distributed via the Huawei Network.

The following documentations are provided for the TOE and delivered by email:

- Installation guide: Huawei EulerOS V2.0 Installation Guide v0.6, delivered in .pdf format.
- User guide: Huawei EulerOS V2.0 User Guide v0.6, delivered in .pdf format.

The list of hardware applicable to the TOE is given above. 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 analyzed and tested:

- Memory separation capability;
- Unavailability of privileged processor states to untrusted user code;
- Full testing of the security functionality on all hardware systems given above;

1.4.2.2 Logical boundaries

All security functions claimed in this ST apply to all architectures and systems allowed via this ST. The primary security features of the TOE include:

- Cryptographic communication: The TOE provides cryptographically secured communication to allow remote entities to log into the TOE. The SSHv2 protocol is provided to set up interactive session with the TOE. The TOE provides both the server side and the client side applications. Using the OpenSSH suite, password-based and public-key-based authentication are allowed.

  The TOE also provides IPSec for a cryptographically secured communication with other remote entities. IPSec is offered together with IKEv1 and IKEv2 for the key negotiation. The implementations of IKEv1 and IKEv2 allow a certificate based authentication of the remote peer.

  The TOE implements TLS protocol to enable a trusted network channel that is used for client and server authentication. It is used in HTTPS service.

- Packet filter: The TOE kernel implements layering structure of network protocols. It has IPTables mechanism to provide a stateful packet filter at network layer and transfer layer for regular IP-based communication. Ethernet frames routed through bridges are controlled by a lower-layer packet filter, EBTables, which is not covered in this evaluation.

- Encrypted data storage: Using tools like dm_crypt or cryptsetup(8), EulerOS can protect block device storage space using cryptography, and the session key used for encryption or decryption can be obtained only with the passphrase. Using Password-Based Key-Derivation Function version 2 (PBKDFv2) implemented with the LUKS mechanism, a user-provided passphrase protects the master volume key which is the symmetric key for encrypting and decrypting data stored on disk. Any data stored on the block devices protected by dm_crypt is encrypted and cannot be decrypted unless the master volume key for the block device is decrypted with the passphrase processed by PBKDFv2. With the device mapper mechanism, the TOE allows for transparent encryption and decryption of data stored on block devices, such as hard disks.

- 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

command like su or sudo. These all rely on explicit authentication information provided interactively by a user.

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

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

- Discretionary Access Control (DAC): DAC allows owners of named objects to control the access permissions to these objects. The owners can permit or deny access by other users based on the configured permission settings. The DAC mechanism is also used to ensure that untrusted users cannot tamper with the TOE mechanisms.

- Auditing: The Lightweight Audit Framework (LAF) is designed to be an audit system making EulerOS 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 to configure the events to be actually audited from the set of all events that are possible to be audited, and to review and search audit logs retrieved.

- Security Management: The security management facilities provided by the TOE are usable by authorized users and/or authorized administrators to modify the configuration of TSF. The TOE allows local management on local consoles and remote management via OpenSSH. Administrative users can log in remotely and perform the same management tasks as a locally operating administrator.

- Boot integrity and system integrity: all components used in booting the OS, including kernel image and kernel modules, are signed by the developer, and are checked for integrity at boot time. Once some integrity violation is found at boot time, the OS kernel will not boot. This is called secure boot. Besides, to ensure system’s runtime integrity, all system sensitive files are protected in user space using IMA appraisal based on hash of the files. If IMA appraisal failed, the access to these key files is then denied by the TOE.

1.4.3 Evaluated configuration

The evaluated configuration is defined as follows:

- The package set evaluated by CC for the TOE must be selected at install time according to the installation guide and be installed accordingly.

- The TOE supports the use of IPv4 and IPv6, both are also supported in the evaluated configuration.

- The default configuration for identification and authentication include both the defined password-based PAM modules and the key-based authentication for OpenSSH. Support for other authentication options, e.g. smart card authentication, is not included in the evaluation configuration.

- If the system console is used, it must be connected directly to the TOE and afforded the same physical protection as the TOE.

- The TOE shall run in the "*Normal mode*" of operation.

Configurations and settings that are different from that specified in the installation guide are not permitted.

Moreover, the TOE was tested on the following physical platforms:

- FusionServer RH2288H V3 Rack Server (with TPM chip embedded).

1.4.4 TOE Environment

A group of TOE systems may be interlinked in a network, and individual networks may be joined by bridges and/or routers, or by TOE systems which act as routers and/or gateways. Each TOE system in the network has its own security policy. The TOE does not include any synchronization mechanism for those policies. As a result, a single user may have user accounts with different UIDs and other different attributes on each of those systems. (A method could be used optionally to synchronize these attributes among the systems in the network, but it is not part of the TOE and must not use methods that conflict with the TOE requirements.)

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

1.4.5 Security Policy Model

The security policy for the TOE is defined by the security functional requirements in chapter 6. The following is a list of the subjects and objects participating in the policy:

Subjects:

- Processes acting on behalf of a human user.

Named Objects:

- User mode file system objects in the following file systems, which reside on persistent storage devices:
  - **Ext4** - standard file system for general data;
  - **XFS** - standard file system for general data;
  - **VFAT** - special purpose file system for UEFI BIOS support mounted at */boot/efi*;
  - **ISO9660** - ISO9660 file system for CD-ROM and DVD;

- Kernel mode file system objects in the following file systems, which reside in kernel memory (which means the file systems are *virtual*):
  - **binfmt_misc** - configuration interface allowing the assignment of executable file formats with user space applications;
  - **cgroup** file system - interface for configuring Linux control groups;
  - **debugfs** - interface for kernel components to provide configuration interfaces to user space;
  - **devpts** – pseudo-terminal file system for allocating virtual TTYs on demand;
  - **devtmpfs** - temporary file system that allows the kernel to generate character or block device nodes;
  - **mqueue** file system - interface for accessing message queues;
  - **procfs** - process file system holding information about processes, general statistical data and tunable kernel parameters;
  - **pstore** file system - interface for accessing persistently stored kernel crash dumps;
  - **rootfs** - the virtual root file system used temporarily during system boot;
  - **selinuxfs** - interface for user space components to interact with the SELinux module inside the kernel, including managing the SELinux policy;
  - **sysfs** - system-related file system covering general information about resources maintained by the kernel including several tunable parameters for these resources;
  - **tmpfs** - the temporary file system backed by RAM;

  Note that the virtual file systems above implement access decisions based on DAC attributes inferred from that of the underlying process. Additional restrictions may apply for specific objects in them.
- Linux system call resources:
  - System call number
  - System call parameter
- Inter Process Communication (IPC) objects:
  - Named pipes
  - Shared memory
  - Message queues
  - Semaphores
  - UNIX domain socket special files

  Note that named pipes and UNIX sockets exist as special files on disks, and access to them are controlled by the same access control mechanism as to other user mode file system objects.
- Network sockets (irrespectively of their type - such as Internet sockets, *netlink* sockets)

TSF data:
- 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 user identifier, group memberships and user password;
- Audit records;
- Volume keys for *dm_crypt* block devices and passphrases protecting the master volume keys;

User data:

- Non-TSF executable code used to drive the behavior of subjects;
- Data not interpreted by TSF and stored or transmitted using named objects;

2 Conformance Claims

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

This Security Target claims conformance to the following specifications:

- Common Criteria for Information Technology Security Evaluation Part 2: Security functional requirements, Version 3.1, Revision 5, September 2012;
- Common Criteria for Information Technology Security Evaluation Part 3: Security assurance requirements Version 3.1, Revision 5, September 2012;
- [OSPP] Common Criteria Protection Profile, BSI-CC-PP-0067, Version 2.0; strict conformance;
- [OSPP-TB] OSPP Extended Package - Trusted Boot, BSI-CC-PP-0067, OSPP EP-TB, Version 2.0; strict conformance;
- [OSPP-IV] OSPP Extended Package - Integrity Verification, BSI-CC-PP-0067, OSPP EP-IV, Version 2.0; strict conformance;
- [OSPP-AM] OSPP Extended Package - Advanced Management, BSI-CC-PP-0067, OSPP EP-AM, Version 2.0; strict conformance;

The Protection Profile and the Extended Packages above referenced can be obtained from the following URL:
https://www.bsi.bund.de/SharedDocs/Zertifikate_CC/PP/aktuell/PP_0067.html

3 Security Problem Definition

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.

The definition of threat agents and protected assets that follows is applicable to the OSPP base, as well as to the OSPP extended packages, unless noted otherwise.

3.1.1 Assets

Assets to be protected are:

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

3.1.2 Threat Agents

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

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

Threat agents are typically characterized by a number of factors, such as expertise, available resources, and motivation, with motivation being linked directly to the value of the assets at stake. The TOE protects against intentional and unintentional breach of TOE security by attackers possessing an enhanced-basic attack potential.

3.1.3 Threats countered by the TOE

The table below lists all threats the TOE can counter:

T.ACCESS.TSFDATA	A threat agent might read or modify TSF data without the necessary authorization when the data is stored or transmitted.
T.ACCESS.USERDATA	A threat agent might gain access to user data stored, processed or transmitted by the TOE without being appropriately authorized according to the TOE security policy.
T.ACCESS.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.ACCESS.COMM	A threat agent might access a communication channel that establishes a trust relationship between the TOE and another remote trusted IT system or masquerade as another remote trusted IT system.
T.RESTRICT.NETTRAFFIC	A threat agent might get access to information or transmit information to other recipients via network communication channels without authorization for this communication attempt by the information flow control policy.
T.IA.MASQUERADE	A threat agent might masquerade as an authorized entity including the TOE itself or a part of the TOE in order to gain unauthorized access to user data, TSF data, or TOE resources.
T.IA.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.ROLE.SNOOP	An attacker might obtain the rights granted to a role that was delegated to another user.
T.ROLE.DELEGATE	An attacker might delegate rights granted to a role that he does not possess or that he is not allowed to delegate.
T.ACCESS.CP.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.ALTER.BOOT	At runtime of the TOE, a threat agent might try to violate the integrity of the TSF code or TSF data loaded and executed before the TSF for integrity verification is active in an undetectable way, potentially enabling bypass of the security policies.
T.ALTER.TSF	A threat agent might try to violate the integrity of the TSF code or TSF data in an undetectable way, resulting in a situation where security policies can be bypassed.
T.ALTER.USERDATA	A threat agent might try to violate the integrity of user data in an undetectable way, resulting in a situation where the TOE cannot reliably store user data.

Table 1 Threats countered by the TOE

3.1.4 Threats to be countered by the TOE environment

TE.MODIFY_ENVIRONMENT	An external entity might try to violate security policies by manipulating the TOE environment; for example, by (directly or indirectly) installing a device driver that uses hardware functions (e.g., direct memory access) to access or violate the integrity of TSF data or TSF functions.

Table 2 Threats to be countered by the TOE environment

3.2 Organizational Security Policies

P.ACCOUNTABILITY	The users of the TOE shall be held accountable for their security-relevant actions within the TOE.
P.USER	Authority shall only be given to users who are trusted to perform the actions correctly.
P.APPROVE	Specific rights assigned to users and controlled by the TSF shall only be exercisable if approved by a second user.
P.PROTECT_SSH_KEY	When using SSH with key-based authentication, organizational procedures must exist that ensure users protect their private SSH key component against its use by any other user.

Table 3 Organizational Security Policies

3.3 Assumptions

The specific conditions below are assumed to exist in a TOE environment.

3.3.1 Physical aspects

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.

Table 4 Assumption: Physical aspects

3.3.2 Personnel aspects

A.MANAGE	The TOE security functionality is managed by one or more competent individuals. The system administrative personnel are not careless, willfully 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.

Table 5 Assumption: Personnel aspects

3.3.3 Procedural aspects

A.DETECT	Any modification or corruption of security-enforcing or security-relevant files of the TOE, user or the underlying platform caused either intentionally or accidentally will be detected by an administrative user.
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.
A.PROTECT.NMENV	It is assumed that the integrity of the following is ensured: • TSF functionality provided with the tamper-protected environment hosting the integrity verification mechanism; • Data used to verify the integrity of the TSF code and TSF data. The first component of the integrity verification TSF functionality used for integrity verification and the integrity verification data for that first component are stored in or provided with that non-modifiable environment.
A.INTEGRITY.TSF.ACTIVE	It is assumed that the integrity verification mechanism provided by the TOE is active for all TSF code and all TSF data at runtime of the TOE.

Table 6 Assumption: Procedural aspects

3.3.4 Connectivity aspects

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.

Table 7 Assumption: Connectivity aspects

4 Security Objectives

4.1 Security Objectives for the TOE

O.AUDITING	The TSF must be able to record defined security-relevant events (which usually include security-critical actions of users of the TOE). The TSF must protect this information and present it to authorized users if the audit trail is stored on the local system. 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 user detect attempted security violations or potential misconfiguration of the TOE security features that would leave the IT assets open to compromise.
O.CRYPTO.NET	The TSF 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 protocols 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.DISCRETIONARY.ACCESS	The TSF must control access of subjects and/or users to named resources based on identity of the object. The TSF must allow authorized users to specify for each access mode which users/subjects are allowed to access a specific named object in that access mode.
O.NETWORK.FLOW	The TOE shall mediate communication between sets of TOE network interfaces, between a network interface and the TOE itself, and between subjects in the TOE and the TOE itself in accordance with its security policy.
O.SUBJECT.COM	The TOE shall mediate communication between subjects acting with different subject security attributes in accordance with its security policy.
O.I&A	The TOE must ensure that users have been successfully authenticated before allowing any action the TOE has defined to provide to authenticated users only.
O.MANAGE	The TSF must provide all the functions and facilities necessary to support the authorized users that are responsible for the management of TOE security mechanisms and must ensure that only authorized users are able to access such functionality.

O.TRUSTED_CHANNEL	The TSF 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.INTEGRITY.BOOT	At runtime of the TOE, the TOE shall verify the integrity of all TSF code, as well as all TSF data loaded and executed before the TSF for integrity verification is active. This objective ensures that the TSF code and TSF data have not been modified when compared to the integrity information in the integrity database.
O.INTEGRITY.BOOT.MGT	The TOE shall be able to allow authorized users to update the integrity verification database covering TSF code and TSF data during maintenance mode of the TOE.
O.INTEGRITY.TSF	The TOE shall be able to verify the integrity of both TSF code and TSF data to ensure that they have not been modified when compared to the integrity information in the integrity database.
O.INTEGRITY.USERDATA	The TOE shall be able to verify the integrity of user data to ensure that it has not been modified when compared to the integrity information in the integrity database.
O.INTEGRITY.ACTION	The TOE shall perform pre-defined actions upon detection of a breach of integrity.
O.INTEGRITY.MANAGE	The TOE shall be able to allow authorized users to update the integrity verification database covering TSF data, the TSF code, and user data. Also, the TOE shall be able to allow authorized users to configure actions to be performed upon the detection of a breach of integrity.
O.ROLE.DELEGATE	The TOE must allow roles assigned to users for performing security-relevant management tasks to be delegated to other users in accordance with the security policy.
O.ROLE.MGMT	The TOE must allow security management actions based on roles to be assigned to different users.
O.ROLE.APPROVE	The TOE must prevent the execution of user actions allowed by a specific right until a second user with a different right approves this action.
O.CP.USERDATA	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.

Table 8 Security Objectives for the TOE

4.2 Security Objectives for the Operational Environment

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.REMOTE	If the TOE relies on remote trusted IT systems to support the enforcement of its policy, those systems provide the functions required by the TOE and are sufficiently protected from any attack that may cause those functions to provide false results.
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 system failure or other discontinuity, recovery without a protection (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.
OE.SECURE_LOAD	The operational environment shall perform checks that ensure the integrity of the TSF code and TSF data loaded and executed before the successful execution of the integrity verification TSF; ensure protection against replay of older versions of that TSF code and TSF data; and ensure that the TSF code and TSF data, when stored, loaded and executed, are protected against reading or loading by unauthorized entities in the TOE environment. If the TOE is started in a different environment or if the TOE is started even when the operational environment has detected a violation of the TOE's integrity, the operational environment shall ensure that the manipulated TOE when started is not able to generate false evidence of its own integrity and the operational environment shall also not falsely generate evidence for the integrity of the TOE when it has not successfully verified the integrity of the TOE.
OE.SECURE_OPERATION	The operational environment shall ensure that the TSF code and TSF data (when in operation) cannot be manipulated or intercepted by entities not under the control of the TOE.

Table 9 Security Objectives for the Operational Environment

4.3 Rationale for Security Objectives

The following tables provide a mapping of security objectives to the environment defined by the threats, policies and assumptions, illustrating that each security objective covers at least one threat, assumption or policy and that each threat, assumption or policy is covered by at least one security objective.

4.3.1 Security Objectives coverage

Objectives	SPD coverage
O.AUDITING	P.ACCOUNTABILITY
O.CRYPTO.NET	T.ACCESS.USERDATA, T.ACCESS.TSFDATA, T.ACCESS.TSFFUNC
O.DISCRETIONARY.ACCESS	T.ACCESS.USERDATA, T.ACCESS.TSFDATA
O.NETWORK.FLOW	T.RESTRICT.NETTRAFFIC
O.SUBJECT.COM	T.ACCESS.USERDATA, T.ACCESS.TSFDATA
O.I&A	T.IA.MASQUERADE, T.IA.USER
O.MANAGE	T.ACCESS.TSFFUNC, P.ACCOUNTABILITY, P.USER

O.TRUSTED_CHANNEL	T.ACCESS.COMM
O.ROLE.DELEGATE	T.ROLE.SNOOP, T.ROLE.DELEGATE
O.ROLE.MGMT	T.ACCESS.TSFFUNC
O.ROLE.APPROVE	P.APPROVE
O.CP.USERDATA	T.ACCESS.CP.USERDATA
O.INTEGRITY.BOOT	T.ALTER.BOOT
O.INTEGRITY.BOOT.MGT	T.ALTER.BOOT
O.INTEGRITY.TSF	T.ALTER.TSF
O.INTEGRITY.USERDATA	T.ALTER.USERDATA
O.INTEGRITY.ACTION	T.ALTER.TSF T.ALTER.USERDATA
O.INTEGRITY.MANAGE	T.ALTER.TSF T.ALTER.USERDATA

Table 10 Coverage of security objectives for the TOE

Objectives	**SPD coverage**
OE.ADMIN	A.AUTHUSER, A.MANAGE, A.TRAINEDUSER
OE.REMOTE	T.ACCESS.COMM, A.CONNECT
OE.INFO_PROTECT	P.USER, A.AUTHUSER, A.TRAINEDUSER, A.PHYSICAL, A.MANAGE P.PROTECT_SSH_KEY
OE.INSTALL	A.MANAGE, A.DETECT A.INTEGRITY.TSF.ACTIVE
OE.MAINTENANCE	A.DETECT
OE.PHYSICAL	A.PHYSICAL A.PROTECT.NMENV
OE.RECOVER	A.MANAGE, A.DETECT
OE.TRUSTED.IT.SYSTEM	A.CONNECT, A.PEER.MGT, A.PEER.FUNC
OE.SECURE_LOAD	A.PROTECT.NMENV
OE.SECURE_OPERATION	TE.MODIFY_ENVIRONMENT

Table 11 Coverage of security objectives for the TOE environment

4.3.2 Security Objectives sufficiency

Threats	**Security Objectives**

T.ACCESS.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.SUBJECT.COM requiring the TSF to mediate communication between subjects.
T.ACCESS.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.SUBJECT.COM requiring the TSF to mediate communication between subjects.
T.ACCESS.TSFFUNC	The threat of accessing TSF functions without proper authorization is removed by: • O.MANAGE requiring that only authorized users utilize management TSF functions. • O.CRYPTO.NET requiring cryptographically-protected communication channels to limit which TSF functions are accessible to external entities. • O.ROLE.MGMT requiring the TOE to allow security management actions based on roles to be assigned to different users.
T.ACCESS.COMM	The threat of accessing a communication channel that establishes a trust relationship between the TOE and another remote trusted IT system is removed 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, • OE.REMOTE requiring that those systems providing the functions required by the TOE are sufficiently protected from any attack that may cause those functions to provide false results.

T.RESTRICT.NETTRAFFIC	The threat of accessing information or transmitting information to other recipients via network communication channels without authorization for this communication attempt is removed by: • O.NETWORK.FLOW requiring the TOE to mediate the communication between itself and remote entities in accordance with its security policy.
T.IA.MASQUERADE	The threat of masquerading as an authorized entity in order to gain unauthorized access to user data, TSF data or TOE resources is removed by: • O.I&A requiring that each entity interacting with the TOE is properly identified and authenticated before allowing any action the TOE is defined to provide to authenticated users only.
T.IA.USER	The threat of accessing user data, TSF data or TOE resources without being identified and authenticated is removed by: • O.I&A 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.ROLE.SNOOP	The threat of an attacker obtaining the rights granted to a role that was delegated to another user is removed by: • O.ROLE.DELEGATE requiring the TOE to allow delegation of roles to other users in accordance with the security policy.
T.ROLE.DELEGATE	The threat of an attacker delegating rights granted to a role that he does not possess or that he is not allowed to delegate is removed by: • O.ROLE.DELEGATE requiring the TOE to allow roles assigned to users for performing security-relevant management tasks to be delegated.
T.ACCESS.CP.USERDATA	The threat of gaining 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 is removed by: • O.CP.USERDATA requiring the TOE to be able to protect the confidentiality of user data at rest separately for each user.
T.ALTER.BOOT	The threat of violating the integrity of the TSF code or TSF data loaded and executed before the TSF for integrity verification at runtime of the TOE is active in an undetectable way is removed by: • O.INTEGRITY.BOOT requiring the TOE to verify the integrity of all TSF code as well as all TSF data loaded and executed before the TSF for integrity verification at runtime of the TOE is active. This objective ensures that the TSF code and TSF data have not been modified when compared to the integrity information in the integrity database.

	• O.INTEGRITY.BOOT.MGT allowing authorized users to update the integrity verification database covering TSF code and TSF data during maintenance mode of the TOE.
T.ALTER.TSF	The threat of violating the integrity of the TSF code or TSF data in an undetectable way, resulting in a situation where security policies can be bypassed is removed by: • O.INTEGRITY.TSF requiring the TOE to verify the integrity of both TSF code and TSF data to ensure that they have not been modified when compared to the integrity information in the integrity database. • O.INTEGRITY.ACTION requiring the TOE to perform pre-defined actions upon detection of a breach of integrity. • O.INTEGRITY.MANAGE requiring the TOE to allow authorized users to update the integrity verification database for TSF data.
T.ALTER.USERDATA	The threat of violating the integrity of the user data in an undetectable way resulting in a situation where the TOE cannot reliably store user data is removed by: • O.INTEGRITY.USERDATA requiring the TOE to verify the integrity of user data to ensure that it has not been modified when compared to the integrity information in the integrity database. • O.INTEGRITY.ACTION requiring the TOE to perform pre-defined actions upon detection of a breach of integrity. • O.INTEGRITY.MANAGE requiring the TOE to allow authorized users to update the integrity verification database for user data.

Table 12 TOE threats sufficiency

TE.MODIFY_ENVIRONMENT	The threat of violating security policies by manipulating the TOE environment is removed by: • OE.SECURE_OPERATION requiring the operational environment to ensure that the TSF code and TSF data (when in operation) cannot be manipulated or intercepted by entities not under the control of the TOE.

Table 13 TOE environment threats sufficiency

Security Policies	**Security Objectives**

P.ACCOUNTABILITY	The policy to hold users accountable for their security-relevant actions within the TOE is implemented by: • O.AUDITING providing the TOE with audit functionality, • O.MANAGE allowing the management of this function.
P.USER	The policy to match the trust given to a user and the actions the user is given authority to perform is implemented by: • O.MANAGE allowing appropriately-authorized users to manage the TSF, • OE.INFO_PROTECT, which requires that users are trusted to use the protection mechanisms of the TOE to protect their data.
P.PROTECT_SSH_KEY	The policy to match the trust given to a user to protect his SSH private key is implemented by: • OE.INFO_PROTECT, which requires that users are trusted to exercise the control over their own data.
P.APPROVE	The policy that specific rights assigned to users shall only be exercisable when approved by a second user is implemented by: • O.ROLE.APPROVE requiring the TOE to prevent the execution of user actions allowed by a specific right until a second user with a different right approves this action.

Table 14 Security policies sufficiency

Assumptions	**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.MANAGE	The assumptions on the TOE security functionality being managed by one or more trustworthy individuals is covered by: • OE.ADMIN requiring trustworthy personnel managing the TOE, • OE.INFO_PROTECT requiring personnel to ensure that information is protected in an appropriate manner, • 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 assure that

	after system failure or other discontinuity, recovery without a protection (security) compromise is achieved.
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 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 requiring competent personnel managing the TOE, • 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 administrative user is covered by: • OE.INSTALL requiring an administrative user to ensure that the TOE is distributed, installed and configured in a secure manner supporting the security mechanisms provided by the TOE, • OE.MAINTENANCE requiring an administrative user to ensure that the diagnostics facilities are invoked at every scheduled preventative maintenance period, verifying the correct operation of the TOE, • OE.RECOVER requiring an administrative user to ensure that procedures and/or mechanisms are provided to assure that after system failure or other discontinuity, recovery without a protection (security) compromise is achieved.
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 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.
A.CONNECT	The assumption on 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.REMOTE requiring that remote trusted IT systems provide the functions required by the TOE and are sufficiently protected from any attack that may cause those functions to provide false results, • OE.TRUSTED.IT.SYSTEM demanding the physical and logical protection equivalent to the TOE.
A.INTEGRITY.TSF.ACTIVE	The assumption that the integrity verification mechanism provided by the TOE is active for all TSF code and all TSF data at runtime of the TOE is covered by: • OE.INSTALL (from OSPP base) requiring proper installation of the TOE ensuring the assumed configuration.

A.PROTECT.NMENV	The assumption that the integrity of the TSF functionality provided with the tamper-protected environment, as well as the TSF data for integrity verification is ensured is covered by: • OE.PHYSICAL (from OSPP base) requiring the physical protection of the execution environment of the TOE. • OE.SECURE_LOAD (from the OSPP Extended Package - Integrity Verification) demanding the proper protection of that TSF code and TSF data.

Table 15 Assumptions sufficiency

5 Extended Components Definition

These extended components are defined in OSPP and its extension packages this ST complies with. They are collected here for easy readability.

5.1 FCS_RNG Generation of random numbers

FCS_RNG.1 Generation of random numbers, requires that the random number generator implements defined security capabilities and that the random numbers meet a defined quality metric.

5.1.1 Family Behaviour

This family defines quality requirements for the generation of random numbers that are intended to be used for cryptographic purposes.

5.1.2 Component leveling:

FCS_RNG.1 is not hierarchical to any other component within the FCS_RNG family.

5.1.3 Management

There are no management activities foreseen.

5.1.4 Audit

There are no actions defined to be auditable.

5.1.5 FCS_RNG.1 Random number generation

Hierarchical to: No other components.
Dependencies: No dependencies.

FCS_RNG.1.1	The TSF shall provide a [**selection**: **deterministic**] random number generator that implements: [**assignment**: • **DRG.2.1: If initialized with a random seed [selection: using PTRNG of class PTG.2 as random source, using PTRNG of class PTG.3 as random source, using NPTRNG of class NTG.1 as random source, [assignment: other requirements for seeding]], the internal state of the RNG shall [selection: have [assignment: amount of entropy], have [assignment: work factor], require [assignment: guess work]]**. • **DRG.2.2: The RNG provides forward secrecy.** • **DRG.2.3: The RNG provides backward secrecy**.].

FCS_RNG.1.2	The TSF shall provide random numbers that meet [**assignment**: • **DRG.2.4: The RNG initialized with a random seed [assignment: requirements for seeding] generates output for which [assignment: number of strings] strings of bit length 128 are mutually different with probability [assignment: probability].** • **DRG.2.5: Statistical test suites cannot practically distinguish the random numbers from output sequences of an ideal RNG. The random numbers must pass test procedure A [assignment: additional test suites]**.].

5.1.6 Rationale

The quality of the random number generator is defined using this SFR. The quality metric required in FCS_RNG.1.2 is detailed in the German Scheme *AIS 20* and *AIS 31*.

5.2 FDP_RIP.3 Full residual information protection of resources

FDP_RIP.3 is analog to FDP_RIP.2 except that it applies to the content of resources that are allocated to subjects or users.

5.2.1 Component leveling

FDP_RIP.3 is not hierarchical to any other component within the FDP_RIP family.

5.2.2 Management

See management description specified for FDP_RIP.2 in [CC].

5.2.3 Audit

See audit requirement specified for FDP_RIP.2 in [CC].

5.2.4 FDP_RIP.3 Full residual information protection of resources

Hierarchical to: No other component
Dependencies: No dependencies

FDP_RIP.3.1	The TSF shall ensure that any previous information content of a resource is made unavailable upon the [**selection: allocation of the resource to, de-allocation of the resource from**] all subjects or users.

5.2.5 Rationale

FDP_RIP.3 addresses the problem of resources implemented in main memory that may be allocated to and de-allocated from subjects or users. Unless those resources lose their content automatically as part of the de-allocation and re-allocation process, they must be subject to a process that prepares them for re-use by rendering the previous content unavailable to the subject or user to which it is next allocated. An example is main memory that has been allocated to a subject; this memory must be cleared before it can be re-allocated to a subject with different security attributes (for example a subject operating on behalf of a different user). This preparation prevents the passing of security-critical information via this resource, since such unregulated passing would potentially allow the subject or user to which the memory is next allocated to use this information to violate the security policy. Typical examples of such critical information that may be passed via resources not prepared for re-use are passwords or cryptographic keys.

5.3 FIA_USB.2 Enhanced user-subject binding

FIA_USB.2 is analog to FIA_USB.1 except that it adds the possibility to specify rules whereby subject security attributes are also derived from TSF data other than user security attributes.

5.3.1 Component leveling

FIA_USB.2 is hierarchical to FIA_USB.1.

5.3.2 Management

See management description specified for FIA_USB.1 in [CC].

5.3.3 Audit

See audit requirement specified for FIA_USB.1 in [CC].

5.3.4 FIA_USB.2 Enhanced user-subject binding

Hierarchical to: FIA_USB.1 User-subject binding
Dependencies: FIA_ATD.1 User attribute definition

FIA_USB.2.1	The TSF shall associate the following user security attributes with subjects acting on the behalf of that user: [**assignment: list of user security attributes**].
FIA_USB.2.2	The TSF shall enforce the following rules on the initial association of user security attributes with subjects acting on the behalf of users: [**assignment: rules for the initial association of attributes**].
FIA_USB.2.3	The TSF shall enforce the following rules governing

	changes to the user security attributes associated with subjects acting on the behalf of users: [**assignment: rules for the changing of attributes**].
FIA_USB.2.4	The TSF shall enforce the following rules for the assignment of subject security attributes not derived from user security attributes when a subject is created: [**assignment: rules for the initial association of the subject security attributes not derived from user security attributes**].

5.3.5 Rationale

An operating system may derive subject security attributes from other TSF data that are not directly user security attributes. An example is the point-of-entry the user has used to establish the connection. An access control policy may also use this subject security attribute within its access control policy, allowing access to critical objects only when the user has connected through specific ports-of-entry.

5.4 FPT_TIM TSF integrity monitoring

FPT_TIM.1 is identical to FDP_SDI.2 defined in the CC except that it applies to TSF and TSF data.

5.4.1 Component leveling

The FPT_TIM family contains only one component: FPT_TIM.1.
FPT_TIM.1 is, therefore, not hierarchical to any other component within the FPT_TIM family

5.4.2 Management

See management description specified for FDP_SDI.2 in [CC].

5.4.3 Audit

See audit requirement specified for FDP_SDI.2 in [CC].

5.4.4 FPT_TIM.1 TSF integrity monitoring and action

Hierarchical to: No other component
Dependencies: No dependencies

FPT_TIM.1.1	The TSF shall monitor [**selection: TSF code, TSF data, [assignment: parts of TSF code, parts of TSF data]] for [assignment: integrity errors] using the following rules: [assignment: rules that define how the integrity is verified**].
FPT_TIM.1.2	Upon detection of a data integrity error, the TSF shall [**assignment: action to be taken**].

5.4.5 Rationale

[CC] defines integrity verification for user data. This SFR extends this functional claim to TSF data and TSF code.

6 Security Requirements

Conforming to OSPP, following typographical conventions are used for marking operations in this ST:

- Assignments and selections are marked in **bold** face font.
- Iterations are marked by appending a suffix to the SFR identification.
- Refinements are marked in bold and italic face font.

6.1 Security Functional Requirements

6.1.1 FAU_GEN.1 Audit data generation

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

a) Start-up and shutdown of the audit functions;
b) All auditable events for the **basic** level of audit; and
c) **all modifications to the set of events being audited;**
d) **all user authentication attempts;**
e) **all denied accesses to objects for which the access control policy defined in the OSPP base applies;**
f) **explicit modifications of access rights to objects covered by the access control policies;**
g) **[assignment: no other auditable events]**

Application Note: FAU_GEN.1.1 has the operations being partially performed to reflect the minimum set of events each operating system conformant to this PP must be able to audit. Since the OSPP base requires that an authorized administrator has the capability to select the events to be audited, all activities that change this set are required to be auditable. In addition, all user authentication attempts must be auditable, but it is allowed that an authorized administrator restricts the events that are actually audited to failed authentication attempts, authentication attempts for specific types of users, authentication attempts when specific authentication methods are used, etc. The rules that allow an authorized administrator to define the events that are actually audited from the set of events the TOE is capable of auditing must be defined in the FAU_SEL.1 (or a hierarchically higher component).

It is also required that the operating system is capable of auditing denied access attempts to objects listed in the access control policies. This requirement allows for analysis of denied access attempts in order to detect a potential misconfiguration of access rights, for example, an attack that performs a large number of access attempts.

Explicit modifications of access rights are those that are performed by an explicit request for access right modification. These are critical if, for example, they are performed by a Trojan Horse.

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

a) Date and time of the event, type of event, subject identity (if applicable), and outcome of the event; and
b) For each audit event type, based on the auditable event definitions of the functional components included in the ST;
   i. **User identity (if applicable); and**
   ii. **[assignment: none]**

Application Note: The subject identity may be identical to the user identity in the case where the subject identity is established by the user-subject binding process. In this case, only one identity needs to be included in the audit record. The purpose here is the ability to trace an event to the user that caused the event. This may not be possible if the subject identity does not allow to identify the user the subject was bound to when the event happened. In order to support FAU_GEN.2, the user identity has, therefore, been added as the information to be recorded.

Application Note: The outcome to be recorded with the audited event can either be binary (success or failure) or the value resulting from the event, depending on the implementation of the TOE. For example, access control decision shall store the information about the result of the access control decision with the audit trail. A TOE may implement more decision results than just access allowed or denied, where all of these results shall be recorded as outcome of the access control check event.

6.1.2 FAU_GEN.2 User identity association

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.

6.1.3 FAU_SAR.1 Audit review

FAU_SAR.1.1 The TSF shall provide **[assignment: the root user]** with the capability to read [**assignment**: **all audit information defined in FAU_GEN.1.2**] 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 FAU_SAR.2 Restricted audit review

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.

6.1.5 FAU_SEL.1 Selective audit

FAU_SEL.1.1 The TSF shall be able to select the set of events to be audited from the set of all auditable events based on the following attributes:

a) **Type of audit event;**
b) **Subject or user identity;**
c) **Outcome (success or failure) of the audit event;**
d) **Named object identity;**
e) **[assignment: Access type to file system objects (read, write, execute, change attributes)**;
f) **System call number]**

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.

6.1.6 FAU_STG.1 Protected audit trail storage

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

FAU_STG.1.2 The TSF shall be able to [**selection**: **prevent]** unauthorized modifications to the audit records in the audit trail.

Application Note: The TOE may store its audit records locally, or it may pass its audit records on to a remote trusted IT system for storage and further processing. Even in this case, the TOE will usually need some kind of local audit trail as a (probably volatile) cache to buffer some audit records or to bridge the time when the remote audit server might not be available. Such a local audit trail must be protected as described in this SFR.

6.1.7 FAU_STG.3 Action in case of possible audit data loss

FAU_STG.3.1 The TSF shall [**assignment**: **notify an authorized administrator**] if the audit trail exceeds [**assignment**: **a root-user selectable, pre-defined size limit of the audit trail**] ***or if any of the following*** **[assignment: no other condition]** ***is detected that may result in a loss of audit records.***

Application Note The term "authorized administrator" refers to the user that is notified by the *auditd* daemon. This daemon can be configured to notify different users in different ways. The administrator of the system must ensure that the *auditd* is configured to send the notification to the intended recipient.

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.

Application Note The information of the threshold limit is done in the configuration file of the *auditd* daemon. This file is only writable to the root user.

6.1.8 FAU_STG.4 Prevention of audit data loss

FAU_STG.4.1 The TSF shall [**selection: overwrite the oldest stored audit records]** and [**assignment: perform one of the following administrator-defined actions:**

a) **Switch to single user mode;**
b) **Halt the system;**

]
if the audit trail is full.

Application Note: The SFR lists all configuration possibilities that apply to the case when the audit trail is full (i.e. the disk is full). Even though the SFR mentions the "ignoring of audit events" separate from the other options, all options should be seen as equal where the root user can select one of these options.

6.1.9 FCS_CKM.1(SYM) Cryptographic key generation

FCS_CKM.1.1 The TSF shall generate ***symmetric*** cryptographic keys in accordance with a specified cryptographic key generation algorithm **capable of generating a random bit sequence** and specified cryptographic key sizes:

a) 128 bits,
b) 168 bits,
c) 256 bits,
d) [**assignment: 192 bits,**
e) **384 bits,**
f) **512 bits,**
]

that meet the following: [**assignment**: **keys are generated based on a random number generator which ensures an entropy that is at least as high as the size of the key and which satisfies the following (based on the use of the key material in cryptographic protocols):**

a) **SSH: generation and exchange of session keys using the Diffie-Hellman key negotiation protocol as defined in RFC4253;**
b) **IPSec: Ciphers for IKE ESP and SA encryption;**
c) **TLS: generation and exchange of session keys as defined in the TLSV1.2 standards with the cipher suites defined in FCS_COP.1(NET);**

]

6.1.10 FCS_CKM.1(RSA) Cryptographic key generation

FCS_CKM.1.1 The TSF shall generate ***RSA*** cryptographic keys in accordance with a specified cryptographic key generation algorithm **defined in U.S. NIST FIPS PUB ~~186-3~~ 186-4 appendix B.3** and specified cryptographic key sizes:

a) 2048 bits,

b) [**assignment**: **3072 bits**
]

that meet the following:

a) **U.S. NIST FIPS PUB ~~186-3~~ *186-4* [1]**,

b) **[assignment: no other standards]**

Application Note: The TOE supports the generation of RSA keys for the OpenSSH host key as well as the OpenSSH user keys using the *ssh-keygen(1)* application. The following random number generator is used to support the key generation:

- FCS_RNG.1(SSL-DFLT) for use in OpenSSH applications;

Application Note: The TOE supports the generation of RSA keys for the IKE and TLS protocols using the *certutil(1)* application. The following random number generator is used to support the key generation:

- FCS_RNG.1(NSS);

6.1.11 FCS_CKM.1(DSA) Cryptographic key generation

FCS_CKM.1.1 The TSF shall generate ***DSA*** cryptographic keys in accordance with a specified cryptographic key generation algorithm **defined in U.S. NIST FIPS PUB ~~186-3~~ 186-4 appendix B.1** and specified cryptographic key sizes:

[**selection**:

a) **L=2048, N=256 bits;**

b) **L=3072, N=256 bits;**

]

that meet the following:

a) **U.S. NIST FIPS PUB ~~186-3~~ *186-4*[2].**

b) [**assignment**: **no other standards**]

Application Note: The TOE supports the generation of DSA keys for the IKE and TLS protocols using the *certutil(1)* application. The following random number generator is used to support the key generation:

- FCS_RNG.1(NSS)

6.1.12 FCS_CKM.1(ECDSA) Cryptographic key generation

FCS_CKM.1.1 The TSF shall generate ***ECDSA*** cryptographic keys in accordance with a specified cryptographic key generation algorithm **defined in U.S. NIST FIPS PUB 186-4 appendix B.4** and specified cryptographic key sizes **defined by the following curves**:

a) **NIST primary field curve P-256;**

b) **NIST primary field curve P-384;**

c) **NIST primary field curve P-521;**

that meet the following:

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

[1] OSPP requires FIPS PUB 186-3, which is superseded by the new version FIPS PUB 186-4.
[2] OSPP requires FIPS PUB 186-3, which is superseded by the new version FIPS PUB 186-4.

Application Note: The TOE supports the generation of ECDSA keys for the OpenSSH host key as well as the OpenSSH user keys using the *ssh-keygen(1)* application. The following random number generator is used to support the key generation:

- FCS_RNG.1(SSL-DFLT) for use in OpenSSH applications;

Application Note: The TOE supports the generation of ECDSA keys for the IKE and TLS protocols using the *certutil(1)* application. The following random number generator is used to support the key generation:

- FCS_RNG.1(NSS)

6.1.13 FCS_CKM.2(NET) Cryptographic key distribution

FCS_CKM.2.1 The TSF shall distribute cryptographic keys in accordance with ***a the following*** specified cryptographic key distribution method ***~~[assignment: cryptographic key distribution method]~~*** that meets the following [**selection**:

a) **Diffie-Hellman key agreement method defined for the SSH protocol by RFC4253: *diffie-hellman-group14-sha1*;**
b) **Public [selection: RSA] host key exchange defined for the SSH protocol by RFC4253;**
c) **RSA encrypted exchange of pre-master secrets defined for the TLS protocol by RFC5246;**
d) **Diffie-Hellman key agreement method defined for the IKEv1 protocol by RFC2409: *modp1024*;**
e) **Diffie-Hellman key agreement method defined for the IKEv2 protocol by RFC5996: *modp1024, modp2048, modp3072, modp4096, modp6144, modp8192*;**
f) [**assignment**: **Diffie Hellman domain parameters or Elliptic Curve reference provided by the remote trusted TLS server;**
g) **Public ECDSA host key exchange defined for the SSH protocol by RFC4253;**
h) **Diffie-Hellman key agreement method defined for the SSH protocol by RFC4253 together with RFC4419: diffie-hellman-group-exchange-sha1, diffie-hellman-group-exchange-sha256;**
i) **EC Diffie-Hellman key agreement method defined for the SSH protocol by RFC4253 together with [RFC5656] : with ecdh-sha2-nistp256, ecdh-sha2-nistp384, ecdh-sha2-nistp521**

]

6.1.14 FCS_CKM.4 Cryptographic key destruction

FCS_CKM.4.1 The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method **of** [**selection**: **zerorization**] that meets the following: [**selection**: **vendor-specific zeroization**].

Application Note: The "vendor-specific zeroization" covers to the following concepts:

- Memory objects: Overwriting the memory with zeros at the time the memory is released.
- Asymmetric key components stored in files: The object reuse functionality for objects defined with FDP_RIP.2 also covers this SFR.

6.1.15 FCS_COP.1(NET) Cryptographic operation

FCS_COP.1.1 The TSF shall perform **encryption, decryption, integrity verification, peer authentication** in accordance with ***a ~~specified~~ the following*** cryptographic algorithms, cryptographic key sizes ***~~[assignment: cryptographic key sizes] that meet the following~~ and applicable standards***: [**selection**:

a) **SSH allowing the use of TDES in CBC mode with 168 bits key size, and HMAC-SHA1 defined by RFC 4253;**
b) **SSH allowing the use of AES in CBC mode with 128 bits and 256 bits key size, and HMAC-SHA1 defined by RFC 4253;**
c) **TLS allowing the use of AES in CBC mode with 128 bits and 256 bits key size, and SHA-1 defined by RFC5246;**
d) **IPSEC with IKE allowing the use of TDES in CTR mode with 168 bits key size, and SHA-1 defined by RFC 4301 and RFC 4303;**
e) **IPSEC with IKE allowing the use of AES in CTR mode with 128 bits and 256 bits key size, and SHA-1 defined by RFC 4301 and RFC 4303;**
f) **[assignment: other cryptographic network algorithms, keys sizes with their standards:**
   - **SSH communication channel encryption using the following ciphers as defined in [RFC4253]:**
     1. **AES in CBC mode (aes192-cbc);**
     2. **AES in CTR mode (aes128-ctr, aes192-ctr, aes256-ctr);**
     3. **AES in GCM mode (aes128-gcm, aes256-gcm);**
     4. **HMAC with SHA-1 (hmac-sha1-etm@openssh.com);**
     5. **HMAC with SHA-2 (hmac-sha2-256, hmac-sha2-512, hmac-sha2-256-etm@openssh.com, hmac-sha2-512-etm@openssh.com) with additional definition in [RFC6668]**
   - **SSH authentication of host as defined in [RFC4252]:**
     1. **RSA signature verification RSASSA-PKCS1-v1.5 using SHA-1 (ssh-rsa)**
     2. **ECDSA with signature verification using SHA-2 (ecdsa-sha2-nistp256 with SHA-256, ecdsa-**

sha2-nistp384 with SHA-384, ecdsa-sha2-nistp521 with SHA-512).

- **SSH authentication of user as defined in [RFC4252]: same ciphers as specified for SSH authentication of host.**
- **IPSEC with IKE the following mechanisms:**
  1. **Ciphers for ESP encryption:**
     i. **AES in CBC mode with 128 bits, 192 bits and 256 bits defined by [RFC3602] supported by [RFC4307].**
     ii. **AES in CTR mode with 192 bits defined by [RFC4301] and [RFC4303].**
  2. **ESP authentication:**
     i. **HMAC SHA-1 truncated to 96 bits;**
  3. **Ciphers for IKE SA encryption:**
     i. **AES in CBC mode with 128 bits, 192 bits and 256 bits defined by [RFC3602] supported by [RFC4307].**
     ii. **AES in CTR mode with 128 bits, 192 bits and 256 bits defined by [RFC4301] and [RFC4303].**
  4. **IKE SA authentication:**
     i. **HMAC SHA-1 truncated to 96 bits;**
  5. **Peer authentication algorithm:**
     i. **RSA**
- **TLS using by the following TLS cipher strings as defined by [RFC5246]**
  1. **Key agreement Diffie-Hellman**
     1) **TLS_DHE_RSA_WITH_AES_128_CBC_SHA**
     2) **TLS_DHE_DSS_WITH_AES_128_CBC_SHA**
     3) **TLS_DHE_RSA_WITH_AES_256_CBC_SHA**
     4) **TLS_DHE_DSS_WITH_AES_256_CBC_SHA**
     5) **TLS_DHE_RSA_WITH_AES_128_CBC_SHA256**
     6) **TLS_DHE_DSS_WITH_AES_128_CBC_SHA256**
     7) **TLS_DHE_RSA_WITH_AES_128_GCM_SHA256**
     8) **TLS_DHE_DSS_WITH_AES_128_GCM_SHA256**
     9) **TLS_DHE_RSA_WITH_AES_256_CBC_SHA256**
     10) **TLS_DHE_DSS_WITH_AES_256_CBC_SHA256**
     11) **TLS_DHE_RSA_WITH_AES_256_GCM_SHA384**
     12) **TLS_DHE_DSS_WITH_AES_256_GCM_SHA384**
  2. **Key agreement EC Diffie-Hellman**
     1) **TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA**
     2) **TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA**
     3) **TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA**
     4) **TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA**
     5) **TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256**
     6) **TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256**
     7) **TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256**
     8) **TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256**
     9) **TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384**

10) **TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384**
11) **TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384**
12) **TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384**

3. **Key exchange RSA**
   1) **TLS_RSA_WITH_AES_128_CBC_SHA**
   2) **TLS_RSA_WITH_AES_128_CBC_SHA256**
   3) **TLS_RSA_WITH_AES_256_CBC_SHA**
   4) **TLS_RSA_WITH_AES_128_GCM_SHA256**
   5) **TLS_RSA_WITH_AES_256_CBC_SHA256**
   6) **TLS_RSA_WITH_AES_256_GCM_SHA384**

]

6.1.16 FCS_COP.1(CP) Cryptographic operation

FCS_COP.1.1 The TSF shall perform **encryption, decryption** in accordance with a specified cryptographic algorithm **formed with any permutation of the following types of cryptographic primitives**:

a) **Ciphers: AES, with key sizes specified in FCS_CKM.1(SYM);**
b) **Block chaining modes: CBC, XTS defined in SP800-38;**
c) **IV-Handling mechanisms:**
   1) **XTS: plain64 - The initialization vector is the 64-bit little-endian version of the sector number, padded with zeros if necessary.**
   2) **CBC: essiv - The sector number is encrypted with the bulk cipher using a salt as key. The salt is derived from the cipher key used for encrypting the data with via hashing using the hashes of either SHA-1, SHA-256, SHA-384 and SHA-512.**
   3) **XTS: benbi - The initialization vector is the 64-bit big-endian version of the sector number, padded with zeros if necessary.**

and cryptographic key sizes **as allowed by the cipher specifications:**

a) **AES: [FIPS197]**
b) **SHA-1 and SHA-2: [FIPS180-4]**

that meet the following: **LUKS-based dm-crypt Linux partition encryption schema**.

Application Note: This SFR applies to the block device disk encryption functionality offered by *dm-crypt*.

6.1.17 FCS_COP.1(IV) Cryptographic operation

FCS_COP.1.1 The TSF shall perform **encryption, decryption, integrity verification** in accordance with a specified cryptographic algorithm:

1) **Hashing the files to be verified using the**

following ciphers: SHA-256

2) **Hashing security related extended attributes based on *evm-key* using the following ciphers: HMAC-SHA-256**
3) **Encrypting *evm-key* by *trusted-key* using the following ciphers: AES256-CBC**

and cryptographic key sizes **256 bits** that meet the following:

1) **Hashing the files to be verified using the following ciphers: [FIPS 180-4]**
2) **Hashing security related extended attributes based on evm-key using the following ciphers: [FIPS 198-1] and [FIPS 180-4]**
3) **Encrypting evm-key by trusted-key using the following ciphers: [NIST SP 800-38A]**

Application Note: The *trusted-key* is generated and stored in TPM. It is loaded into OS kernel and used to protect *evm-key*. Evm-key is saved to disk after encryption and loaded into kernel at boot time.

6.1.18 FCS_RNG.1(SSL-DFLT) Random number generation (Class DRG.2)

FCS_RNG.1.1 The TSF shall provide a **deterministic** random number generator that implements:

a) **DRG2.1: If initialized with a random seed using /dev/random as random source, the internal state of the RNG shall have a min-entropy of 48 bits.**
b) **DRG2.2: The DRNG provides forward secrecy.**
c) **DRG2.3: The DRNG provides backward secrecy.**

FCS_RNG.1.2 The TSF shall provide random numbers that meet:

a) **DRG.2.4: The RNG instance initialized with a random seed**
   1. **every time the ssh client is invoked**
   2. **every time the ssh-keygen application is invoked**
   3. **every time the sshd server processes a new connection**

   generates output for which 219 strings of bit length 128 are mutually different with probability of more than 1 - (2**(-10)).**
b) **DRG.2.5: Statistical test suites cannot practically distinguish the random numbers from output sequences of an ideal RNG. The random numbers must pass test ~~procedures~~ *suite* A [assignment: no other test suite].**

Application Note: The *OpenSSH* application uses the deterministic RNG (default mode) from *OpenSSL* to generate random numbers. Every time the *ssh* client is invoked, or the *ssh-keygen* application is used, or a new SSH connection is processed by *sshd*, the deterministic random number generator is seeded with data from /dev/random.

Application Note: FIPS mode is enabled for the *openssl* library, and this also impacts *OpenSSH*.

6.1.19 FCS_RNG.1(DM-INIT) Random number generation (Class DRG.2)

FCS_RNG.1.1 The TSF shall provide a **deterministic** random number generator that implements:

a) **DRG2.1: If initialized with a random seed using high-resolution time stamps of block device access events, human interface device events and interrupt events, the internal state of the RNG shall have a min-entropy of 48 bits.**

b) **DRG2.2: The DRNG provides forward secrecy.**

c) **DRG2.3: The DRNG provides backward secrecy**.

FCS_RNG.1.2 The TSF shall provide random numbers that meet:

a) **DRG.2.4: The RNG instance initialized with a random seed during initialization of the cryptsetup application generates output for which 2**19 strings of bit length 128 are mutually different with probability of more than 1- (2**(-10)).**

b) **DRG.2.5: Statistical test suites cannot practically distinguish the random numbers from output sequences of an ideal RNG. The random numbers must pass test ~~procedures~~ *suite* A [assignment: no other test suite].**

6.1.20 FCS_RNG.1(DM-RUN) Random number generation (Class DRG.2)

FCS_RNG.1.1 The TSF shall provide a **deterministic** random number generator that implements:

a) **DRG2.1: If initialized with a random seed using /dev/random as random source, the internal state of the RNG shall have a min-entropy of 48 bits.**

b) **DRG2.2: The DRNG provides forward secrecy.**

c) **DRG2.3: The DRNG provides backward secrecy**.

FCS_RNG.1.2 The TSF shall provide random numbers that meet:

a) **DRG.2.4: The RNG instance initialized with a random seed during initialization of the cryptsetup application generates output for which 2**19 strings of bit length 128 are mutually different with probability of more than 1 - (2**(-10)).**

b) **DRG.2.5: Statistical test suites cannot practically distinguish the random numbers from output sequences of an ideal RNG. The random numbers must pass test ~~procedures~~ *suite* A [assignment: no other test suite].**

6.1.21 FCS_RNG.1(NSS) Random number generation (Class DRG.2)

FCS_RNG.1.1 The TSF shall provide a **deterministic** random number generator that implements:

a) **DRG2.1: If initialized with a random seed using high-resolution time stamps of block device access events, human interface device events and interrupt events, the internal state of the RNG shall have a min-entropy of 48 bits.**
b) **DRG2.2: The DRNG provides forward secrecy.**
c) **DRG2.3: The DRNG provides backward secrecy.**

FCS_RNG.1.2 The TSF shall provide random numbers that meet:

a) **DRG.2.4: The RNG instance initialized with a random seed during initialization of the *cryptsetup* application generates output for which 2**19 strings of bit length 128 are mutually different with probability of more than 1 - (2**(-10)).**
b) **DRG.2.5: Statistical test suites cannot practically distinguish the random numbers from output sequences of an ideal RNG. The random numbers must pass test ~~procedures~~ *suite* A [assignment: no other test suite].**

Application Note: The NSS library uses an SP800-90A Hash DRBG with SHA-256 core using a derivation function without prediction resistance.

6.1.22 FCS_RNG.1(IV) Random number generation (Class DRG.2)

FCS_RNG.1.1 The TSF shall provide a **deterministic** random number generator that implements:

d) **DRG2.1: If initialized with a random seed using /dev/random as random source, the internal state of the RNG shall have a min-entropy of 48 bits.**
e) **DRG2.2: The DRNG provides forward secrecy.**
f) **DRG2.3: The DRNG provides backward secrecy.**

FCS_RNG.1.2 The TSF shall provide random numbers that meet:

c) **DRG.2.4: The RNG instance initialized with a random seed during startup of the keyctl application generates output for which 2**19 strings of bit length 128 are mutually different with probability of more than 1 - (2**(-10)).**
d) **DRG.2.5: Statistical test suites cannot practically distinguish the random numbers from output sequences of an ideal RNG. The random numbers must pass test ~~procedures~~ *suite* A [assignment: no other test suite].**

6.1.23 FDP_ACC.1(PSO) Subset access control

FDP_ACC.1.1 The TSF shall enforce the **Persistent Storage Object Access Control Policy** on:

a) [**assignment**: **Subjects: all subjects defined with the Security Policy Model;**]
b) **Objects:**

i. **Persistent Storage Objects of the following type:** [**assignment**: **all file system objects defined with the Security Policy Model**];
ii. [**assignment**: **no other storage objects;**]

c) **Operations**: [**assignment**: **read, write, execute (regular files), search (directories)**].

Application Note: A persistent storage object establishes a data storage or data exchange link between two or more subjects. Examples of persistent storage objects are: files, directories.

Application Note: Not all operations listed are supported for all persistent objects. E.g., WRITE to a file or directory in an ISO9660 file system is always denied. However, once an operation is supported for the object, it is always controlled by the security policy.

Application Note: Although a persistent file system, *vfat* is not POSIX-compliant and hence does not support the access control policy for each single file/directory. However, the security attributes supporting the access control policy can be specified uniformly for all files/directories *as a whole* at mount time, such as uid=uuu,gid=ggg,fmask=0000,dmask=0000. As a result, command /bin/chown always fails for vfat objects, and command /bin/chmod seems to work well on vfat objects, but it does not change anything.

6.1.24 FDP_ACC.1(TSO) Subset access control

FDP_ACC.1.1 The TSF shall enforce the **Transient Storage Object Access Control Policy** on:

a) [**assignment**: **Subjects: all subjects defined with the Security Policy Model;**]

b) **Objects:**

i. **Transient Storage Objects of the following type: [assignment: all IPC objects defined with the Security Policy Model;]**
ii. **[assignment: no other storage objects;]**

c) **Operations:** [**assignment**: **read, receive, write, send**].

Application Note: A transient storage object establishes a data exchange link between two or more subjects or users. Examples of transient storage objects are: shared memory, semaphores, message queues, named/unnamed pipes.

6.1.25 FDP_ACF.1(PSO) Security attribute based access control

FDP_ACF.1.1 The TSF shall enforce the **Persistent Storage Object Access Control Policy** to objects based on the following: [**assignment**:

a) **Subject security attributes: file system UID, file system GID, supplementary GIDs;**

b) **Object security attributes: owning UID, owning GID;**

c) **Access control security attributes maintained for each file system object governing access to that object:**

i. **Permission bits for the owning UID,**
ii. **Permission bits for the owning GID,**
iii. **Permission bits for all other users (“world”),**
iv. **The following permission bits: read, write, execute (for files), search (for directories),**
v. **The following access rights applicable to the file system object: immutable (files),**

d) **Access control security attributes maintained for each disk partition that holds a file system: read-only, no-execute**

]

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

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 one of the following rules hold:

a) **The subject's filesystem UID is identical with the owning UID of the object and the requested type of access is within the permission bits defined for the owning UID (permission bits); or**
b) **The subject's filesystem GID or one of the subject's supplementary GIDs is identical with the owning GID and the requested type of access is within the permission bits defined for the owning GID (permission bits); or**
c) **The requested type of access is within the permission bits defined for "all other users" (permission bits).**

]

FDP_ACF.1.3 The TSF shall explicitly authorise access of subjects to objects based on the following additional rules: [**assignment**:

a) **read and directory search operations are allowed for the subject with the capability of CAP_DAC_READ_SEARCH;**
b) **write and execute operations are allowed for the subject with the capability of CAP_DAC_OVERRIDE - the execute permission is granted if the file system object is marked with at least one executable bit in its permission settings.**

]

FDP_ACF.1.4 The TSF shall explicitly deny access of subjects to named objects based on the following rules: [**assignment**:

a) **Any file system object in a file system that is mounted as read-only cannot be modified, created or removed;**
b) **A regular file, a directory and a symbolic link in a file system that is mounted as read-only cannot be written to;**
c) **Any file system object marked as immutable cannot be modified or removed;**
d) **A regular file in a file system that is mounted with the no-execute flag cannot be executed;**
e) **Any file system object stored in a directory marked with the SAVETXT bit cannot be modified or removed by subjects whose file system UID is not equal to the owning UID of the file system object unless the subject performing the operation possesses the CAP_FOWNER capability.**

]

Application note: Although a persistent file system, *vfat* is not POSIX-compliant and hence does not support the access control policy for each single file/directory. However, the security attributes supporting the access control policy can be specified uniformly for all files/directories *as a whole* at mount time, such as uid=uuu,gid=ggg,fmask=0000,dmask=0000. As a result, command /bin/chown always fails for vfat objects, and command /bin/chmod seems to work well on vfat objects, but it does not change anything.

6.1.26 FDP_ACF.1(TSO) Security attribute based access control

FDP_ACF.1.1 The TSF shall enforce the **Transient Storage Object Access Control Policy** to objects based on the following: [**assignment**:

a) **Subject security attributes: effective UID, file system UID, effective GID, file system GID, supplementary GIDs;**
b) **Object security attributes: owning UID, owning GID;**
c) **Access control security attributes maintained for each IPC object whose name is managed with a file governing access to that object: see FDP_ACF.1(PSO);**
d) **Access control security attributes maintained for any other IPC object governing access to that object:**
    i. **Permission bits for the owning UID;**
    ii. **Permission bits for the owning GID;**
    iii. **Permission bits for "world";**
    iv. **The following permission bits: read, write, execute**

]

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

a) **IPC object whose name is managed with a file: see FDP_ACF.1(PSO);**
b) **Any other IPC object: A subject has a specific type access to an object if one of the following rules hold:**
   1. **The subject's effective UID is identical with the owning UID of the object and the requested type of access is within the permission bits defined for the owning UID; or**
   2. **The subject's effective GID or one of the subject's supplementary GIDs is identical with the owning GID and the requested type of access is within the permission bits defined for the owning GID; or**
   3. **The requested type of access is within the permission bits defined for "world".**

]

FDP_ACF.1.3 The TSF shall explicitly authorise access of subjects to objects based on the following additional rules: [**assignment**:

a) **IPC object whose name is managed with a file: see FDP_ACF.1(PSO);**
b) **Any other IPC object:**
   1. **read, write, send and receive operations are allowed for the subject with the capability of CAP_IPC_OWNER.**

]

FDP_ACF.1.4 The TSF shall explicitly deny access of subjects to named objects based on the following rules: [**assignment**:

a) **IPC object whose name is managed with a file: see FDP_ACF.1(PSO);**
b) **Any other IPC object: none.**

]

6.1.27 FDP_IFC.2(NI) Complete information flow control

FDP_IFC.2.1 The TSF shall enforce the **Network Information Flow Control Policy** on:

a) **Subjects:**
   i. **unauthenticated external IT entities that send and receive information mediated by the TOE;**
   ii. **[assignment: standard Linux processes] that send and receive information mediated by the TOE;**
b) **Information:**
   i. **Network data routed through the TOE;**
   ii. **[assignment: Network data received by the TOE from an external IT entity;**
   iii. **Network data provided to the TOE by a subject executing on the TOE intended to be sent to an external IT entity via a network interface controlled by the TOE**

   ]

and all operations that cause that information to flow to and

from subjects covered by the SFP.

FDP_IFC.2.2 The TSF shall ensure that all operations that cause any information in the TOE to flow to and from any subject in the TOE are covered by an information flow control SFP.

Application Note: The OSPP explicitly does not specify the version of the Internet Protocol. This implies that the Internet Protocol versions usable in the evaluated configuration must be covered by this SFR.

6.1.28 FDP_IFF.1(NI) Simple security attributes

FDP_IFF.1.1 The TSF shall enforce the **Network Information Flow Control Policy** based on the following types of subject and information security attributes:

a) **~~Object~~*Information* security attribute: the logical or physical network interface through which the network data *from an external IT entity* entered the TOE *or is intended to be sent out;***

[selection:

b) **TCP/IP information security attributes:**
   i. **Source and destination IP address,**
   ii. **Source and destination TCP port number,**
   iii. **Source and destination UDP port number,**
   iv. **Network protocol of [selection: TCP, UDP, ICMP, [assignment: no other protocols]]**
   v. **TCP header flags of [selection: SYN, ACK, [assignment: FIN, RST, URG, PSH, TCP sequence numbers]],**
   vi. **[assignment: TCP sequence numbers]**

]

Application Note: Logical network interfaces include the interface provided by the TOE to local subjects acting on behalf of local users. Such interfaces may include network sockets introduced by the Berkeley Software Distribution (BSD) or any other mechanism that allows subject to initiate an IP-based connection.

Application Note: The minimum requirement of the network flow control specified in FDP_IFF.1.3(NI) defines the purpose of the Network Information Flow Control Policy, namely to identify network data using the security attributes specified here and to at least discard the identified network data or allow it to pass the TOE unaltered. An ST author may specify network data security attributes which differ from the TCP/IP attributes or the VLAN tag attributes. However, these security attributes must allow the minimum requirements of FDP_IFF.1.3(NI) to be achieved.

FDP_IFF.1.2 The TSF shall permit an information flow between a controlled subject and controlled information via a controlled operation if the following rules hold: [**assignment**:

a) **If the set of rules defined in accordance with the security attributes defined in FDP_IFF.1.3 define that the network data is discarded, the network data shall**

not be delivered by the TOE to the intended recipient;

b) **If the set of rules defined in accordance with the security attributes defined in FDP_IFF.1.3 define that the network data is to be delivered unaltered, the network data shall be delivered unaltered by the TOE to the intended recipient;**

c) **If the set of rules defined in accordance with the security attributes defined in FDP_IFF.1.3 define another action to be taken than discarding the network data or delivering the data unaltered to the intended recipient, the TOE shall perform this action.**

]

FDP_IFF.1.3 The TSF shall enforce **the following rules:**
Identification of network data using one or more of the following concepts:

a) **Information security attribute matching *based on the following security attributes:***
   1. ***IP header information,***
   2. ***UDP header information,***
   3. ***TCP header information,***
   4. ***ICMP type and code,***
   5. ***incoming network interface,***
   6. ***outgoing network interface***

b) **[selection: Matching based on the state of a TCP connection, [assignment: no other matching concepts]];**

Performing one or more of the following actions with identified network data:

a) **Discard the network data [selection: without any further processing, with sending a notification to the sender];**

b) **Allow the network data to be processed unaltered by the TOE according to the routing information maintained by the TOE;**

c) **[assignment: no other actions].**

FDP_IFF.1.4 The TSF shall explicitly authorize an information flow based on the following rules: [**assignment**:
If the network data is not matched by the rule set and the default rule of the packet filter is ACCEPT then the data is forwarded unaltered based on the normal operation of the host system's networking stack.]

FDP_IFF.1.5 The TSF shall explicitly deny an information flow based on the following rules: [**assignment**:
If the network data is not matched by the rule set, one of the following default rules applies:

a) **DROP: the data is discarded.**

]

Application Note: The OSPP explicitly does not specify the version of the Internet Protocol. This implies that the Internet Protocol versions usable in the evaluated configuration must be covered by this SFR.

Application Note: The default rule is configurable where exactly one of the above mentioned default rules can be selected at any given time.

Application Note: The SFRs FDP_IFF.1(NI) defines different rule sets implemented by the TOE covering the FDP_IFF.1 SFR from the OSPP base.

6.1.29 FDP_ITC.2 Import of user data with security attributes

FDP_ITC.2.1 The TSF shall enforce the **Persistent Storage Access Control Policy**, **Transient Storage Access Control Policy**, **Network Information Flow Control**, **[assignment: no other access control SFP(s) and/or information flow control SFP(s)]** when importing user data, controlled under the SFP, from outside of the TOE.

FDP_ITC.2.2 The TSF shall use the security attributes associated with the imported user data.

FDP_ITC.2.3 The TSF shall ensure that the protocol used provides for the unambiguous association between the security attributes and the user data received.

FDP_ITC.2.4 The TSF shall ensure that interpretation of the security attributes of the imported user data is as intended by the source of the user data.

FDP_ITC.2.5 The TSF shall enforce the following rules when importing user data controlled under the SFP from outside the TOE: [**assignment**: **No additional importation control rules.]**

Application Note: If, for example, file names or file name extensions are used for access control decisions, they are security attributes. In the case that an external file system is mounted, this is considered an import of user data with security attributes, and therefore, FDP_ITC rules must be defined and satisfied.

Application Note: Based on the wording of FDP_ITC.2.1, the TOE complies with this SFR even when it does not allow import of objects covered by the persistent or transient storage object control policy.
However, the network information flow control policy must always be covered by the TOE, as it applies to the networking capability of the TOE to control traffic originating from outside the TOE. In this case, the interpretation of security attributes is defined by the respective protocol family.

6.1.30 FDP_RIP.2 Full residual information protection

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

6.1.31 FDP_RIP.3 Full residual information protection of resources

FDP_RIP.3.1 The TSF shall ensure that any previous information content of a resource is made unavailable upon the [**selection**: **allocation of the resource to**] all subjects or users.

6.1.32 FIA_AFL.1 Authentication failure handling

FIA_AFL.1.1 The TSF shall detect when **an administrator-configurable number of** unsuccessful authentication attempts ***for the authentication method [assignment: of password-based authentication]*** occur related to [**assignment**: **consecutive unsuccessful authentication attempts]**.

FIA_AFL.1.2 When the defined number of unsuccessful authentication attempts has been [**selection**: **met]**, the TSF shall [**assignment:**

a) **For all accounts, "disable" the account for a time period configured by the root user.**
b) **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 TOE may use different authentication methods for different types of users and have different rules for how to handle authentication failures based on the authentication method and/or user type. Authentication failures for remote systems are usually treated differently from authentication attempts for human users. Even for human users, the reaction to authentication failures may be different for authentication via userid/password and authentication via smartcards or digital certificates.

6.1.33 FIA_ATD.1(HU) User attribute definition

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

a) **User identifier;**
b) **Group memberships;**
c) **User password;**
d) **Software token verification data;**
e) **Security roles;**
f) [**assignment**: **no other user security attributes**]

Application Note: See the application note for FIA_UAU.5 for a list of token-based authentication mechanisms and their associated tokens.

6.1.34 FIA_ATD.1(TU) User attribute definition

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

a) **the logical or physical network interface through which the network data entered the TOE;**
b) **identity of the logical or physical external interface through which the user connected to the TOE;**
c) **[assignment: no other user security attributes]**

Application Note: Bullet a) of this SFR relates to FDP_IFC.2(NI) and FDP_IFF.1(NI). In the Common Criteria scheme, external entities are always considered to be users. Therefore, every network data entity must be specified as user in this PP.

6.1.35 FIA_SOS.1 Verification of secrets

FIA_SOS.1.1 The TSF shall provide a mechanism to verify that secrets meet **the following quality metric: the probability that a secret can be obtained by an attacker during the lifetime of the secret is less than 2^(-20).**

Application Note: The TOE password change is implemented using the PAM library. The PAM module *pam_pwquality.so* allows the specification of the quality of new passwords. The evaluated configuration requires a configuration of the PAM-based password change mechanism that meets the above mentioned criteria.

Application Note: For key-based authentication methods, the evaluation of the RSA, DSA, and ECDSA keys used for the SSH protocol will show the maximum lifetime of a key depending on its size.

6.1.36 FIA_UAU.1 Timing of authentication

FIA_UAU.1.1 The TSF shall allow

a) **the information flow covered by the Network Information Flow Control Policy;**
b) **[assignment: Establishing a cryptographically secured network connection;**
c) **Local console log-in: banner information;**
d) **SSH log-in: obtaining the list of allowed authentication methods;**
]

on behalf of the user to be performed before the user is authenticated.

Application note The banner for local login is in file /etc/issue. The banner for remote login is default to file /etc/issue.net and can be set in file /etc/ssh/sshd_config.

FIA_UAU.1.2 The TSF shall require each user to be successfully authenticated before allowing any other TSF-mediated actions on behalf of that user.

6.1.37 FIA_UAU.5 Multiple authentication mechanisms

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

a) **Authentication based on username and password;**
b) **Authentication based on software token verification data;**
c) **[assignment: no other authentication**

mechanisms]

to support user authentication.

Application Note: The TOE is able to maintain the following types of software tokens and their 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. When the TOE acts as an SSH client, the TOE is able to store the private part of the SSH user key for the requesting user.

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;**
c) [**assignment**: **Users with expired passwords are required to create a new password after correctly entering the expired password;**
d) **For SSH, both the password-based and key-based authentication methods can be enabled at the same time. In this case, the key-based authentication method is tried before the password-based authentication. If the key-based authentication succeeds, the user is authenticated. If the 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 the term “software token verification data”, see the application note for FIA_ATD.1(HU).

6.1.38 FIA_UAU.7 Protected authentication feedback

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

6.1.39 FIA_UID.1 Timing of identification

FIA_UID.1.1 The TSF shall allow [**assignment**:

a) **the information flow covered by the Network Information Flow Control Policy;**
b) **Establishing a cryptographically secured network connection;**
c) **Console log-in: banner information;**
d) **SSH log-in: obtaining the list of allowed authentication methods;**

]

on behalf of the user to be performed before the user is identified.

FIA_UID.1.2 The TSF shall require each user to be successfully identified before allowing any other TSF-mediated actions on behalf of that user.

6.1.40 FIA_USB.2 Enhanced user-subject binding

FIA_USB.2.1 The TSF shall associate the following user security attributes with subjects acting on the behalf of that user:

a) **The user identity that is associated with auditable events;**
b) **The user security attributes that are used to enforce the Persistent Storage Object Access Control Policy;**
c) **The user security attributes that are used to enforce the Transient Storage Object Access Control Policy;**
d) **The software token that can be used for subsequent identification and authentication with the TSF or other remote IT systems;**
e) **Active roles;**
f) **Active groups;**
g) [**assignment**: **no other security attributes**]

FIA_USB.2.2 The TSF shall enforce the following rules on the initial association of security attributes with subjects acting on the behalf of users: [**assignment**:

a) **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;**
b) **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;**
c) **Upon successful identification and authentication, the supplementary GIDs shall be those specified via the supplementary group membership assignment for the user entry;**

]

Application Note: The various subject UIDs are all derived from the same numeric UID per user entry stored in the */etc/passwd* file. The subject's supplementary GIDs are derived from the username to group name mappings in the */etc/group* file. As the TOE only maintains numeric IDs for subjects, the username and the group names need to be converted before instantiating the subject. The username to UID mapping is provided in */etc/passwd* file and the group name to GID mapping is provided in */etc/group* file.

FIA_USB.2.3 The TSF shall enforce the following rules governing changes to the user security attributes associated with subjects acting on the behalf of users: [**assignment**:

a) **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.

b) **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.**

c) **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 processes possessing the *CAP_SETUID* capability.**

d) **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 processes possessing the *CAP_SETGID* capability.**

e) **The set of supplementary GIDs of a subject can be changed by the use of the *setgroups* system call for the calling application. These system calls are restricted to processes possessing the *CAP_SETGID* capability.**

f) **The set of effective and inheritable capabilities of a subject can be changed by the use of an executable with activated file capabilities. In this case the program obtains the following capabilities when invoking the file with *execve*:**

1. **the process' the effective capability set gains the capabilities defined by the permitted file capabilities set;**
2. **the process' inheritable capability set is ANDed with the inheritable file capability set to form the new process' inheritable capability set which defines the capability set that will be retained after an *execve* system call.**

]

FIA_USB.2.4 The TSF shall enforce the following rules for the assignment of subject security attributes not derived from user security attributes when a subject is created:
[assignment: no rules].

Application Note: The applications "su" and "sudo" allow the calling user to change the filesystem
and effective UID either to root or to other users provided the authentication to "su" or "sudo" was successful. Both application uses the SETUID bit with the owning UID of root as well as the set*uid system calls to change to other UIDs before

spawning a new shell or the given command. As both applications rest on the above mentioned mechanisms, it is not listed as a separate mechanism to modify the calling user's UIDs.

Application Note: The login UID is set by the PAM modules by inserting the intended UID into the */proc/<PID>/loginuid* file. This file can be written to only by subjects executing with the effective UID of zero (root) and only for the calling process' own *loginuid* file. However, there is no application except the PAM modules which access that proc file which implies that the login UID remains unchanged after login when operating the TOE. Authorized administrators are not intended to access that proc file.

6.1.41 FMT_MSA.1(PSO) Management of object security attributes

FMT_MSA.1.1 The TSF shall enforce the **Persistent Storage Object Access Control Policy** to restrict the ability to **modify**, [**selection**: **change_default]** the security attributes **of the objects covered by the SFP** to **the owner of the object and** [**assignment**: **users with processes granted the *CAP_CHOWN*, *CAP_FOWNER*, *CAP_FSETID* capabilities**].

6.1.42 FMT_MSA.1(TSO) Management of object security attributes

FMT_MSA.1.1 The TSF shall enforce the **Transient Storage Object Access Control Policy** to restrict the ability to **modify**, [**selection: change_default]** the security attributes **of the objects covered by the SFP** to **the owner of the object and** [**assignment**: **users with processes granted the *CAP_SYS_ADMIN* capability**].

6.1.43 FMT_MSA.3(PSO) Static attribute initialization

FMT_MSA.3.1 The TSF shall enforce the **Persistent Storage Object 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 [**assignment**:

a) **administrators for a global setting applied during logon;**

b) **users for a setting applicable to his processes;**

]

to specify alternative initial values to override the default values when an object
or information is created.

Application Note: The global 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 the root user can manage that initial value as this file is writable to root only. Users can change their umask value at any time using the *umask(2)* system call.

6.1.44 FMT_MSA.3(TSO) Static attribute initialization

FMT_MSA.3.1 The TSF shall enforce the **Transient Storage Object 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 [**assignment**:

a) **administrator for a global setting applied during logon;**

b) **users for a setting applicable to his processes**

]

to specify alternative initial values to override the default values when an object
or information is created.

Application Note: The global 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 the root user can manage that initial value as this file is writable to root only. Users can change their *umask* value at any time using the *umask(2)* system call.

NOTE: *umask* does not impact on System V IPC objects, and the permission bits for them is given explicitly at creation time.

6.1.45 FMT_MSA.3(NI) Static attribute initialization

FMT_MSA.3.1 The TSF shall enforce the **Network Information Flow Control Policy** to provide [**selection**: **permissive**] default values for security attributes that are used to enforce the SFP.

FMT_MSA.3.2 The TSF shall allow [**assignment**:
a) users with the administrator user role;
b) users with processes granted the CAP_NET_ADMIN capability] to specify alternative initial values to override the default values when an object or information is created

Application Note: The default value specified in this SFR applies to the default for the packet filter after boot. The administrator can configure alternative default values as outlined in FDP_IFF.1(NI).

Application Note: The *iptables* command uses a *netlink* interface to the kernel which requires that the caller possesses the *CAP_NET_ADMIN* capability.

6.1.46 FMT_MSA.4(PSO) Security attribute value inheritance

FMT_MSA.4.1 The TSF shall use the following rules to set the value of security attributes ***for Persistent Storage Objects*:** [**assignment**:

a) **The newly created object's owning UID is set to the effective UID of the calling subject;**

b) **The newly created object's owning GID is set to the effective GID of the calling subject with the following exception for file system objects: if the parent directory holding the newly created file system object**

is marked with the *SETGID* permission bit, the owning GID of the newly created file system object is set to the owning GID of the parent directory;

c) **The newly created object's permission bits are derived from the calling subject's *umask* value by masking out the *umask* bits from the permission bit set granting full access;**

]

6.1.47 FMT_MTD.1(AE) Management of TSF data

FMT_MTD.1.1 The TSF shall restrict the ability to **query, modify** the **set of audited events** to [**assignment**: **administrators**].

Application Note: This SFR applies to FAU_SEL.1

6.1.48 FMT_MTD.1(AS) Management of TSF data

FMT_MTD.1.1 The TSF shall restrict the ability to **clear,** [**selection**: **configure the storage location, delete**] the audit storage to [**assignment: administrators**].

Application Note: This SFR applies to FAU_STG.1 where the directory used for storing the audit trail is configured.

Application Note: The configuration of these parameters is performed with the configuration file */etc/auditd/auditd.conf* which is writable to the root user only.

6.1.49 FMT_MTD.1(AT) Management of TSF data

FMT_MTD.1.1 The TSF shall restrict the ability to **modify,** [**selection: add, delete**] the

a) **threshold of the audit trail when an action is performed;**

b) **action when the threshold is reached**

to [**assignment: administrators**].

Application Note: This SFR applies to FAU_STG.3.

Application Note: The configuration of these parameters is performed with the configuration file */etc/auditd/auditd.conf* which is writable to the root user only.

6.1.50 FMT_MTD.1(AF) Management of TSF data

FMT_MTD.1.1 The TSF shall restrict the ability to **modify,** [**selection: add, delete**] the **actions to be taken in case of audit storage failure** to [**assignment**: **administrators**].

Application Note: This SFR applies to FAU_STG.4.

Application Note: The configuration of these parameters is performed with the configuration file */etc/auditd/auditd.conf* which is writable to the root user only.

6.1.51 FMT_MTD.1(NI) Management of TSF data

FMT_MTD.1.1 The TSF shall restrict the ability to **query, modify, delete,** [**selection**: **change_default**] the **security attributes for the rules governing the**

a) **identification of *and matching of*[3] network data;**

b) **actions performed on the identified network data;**

to [**assignment**: **administrators**].

Application Note: This SFR applies to FDP_IFF.1(NI).

6.1.52 FMT_MTD.1(IAT) Management of TSF data

FMT_MTD.1.1 The TSF shall restrict the ability to **modify** the **threshold for unsuccessful authentication attempts** to [**assignment**: **administrators**].

Application Note: This SFR applies to FIA_AFL.1.

Application Note: The configuration of these parameters is performed with the PAM configuration files which are writable to the root user only.

6.1.53 FMT_MTD.1(IAF) Management of TSF data

FMT_MTD.1.1 The TSF shall restrict the ability to **re-enable** the **authentication to the account subject to authentication failure** to [**assignment**: **administrators**].

Application Note: This SFR applies to FIA_AFL.1.

Application Note: The account locking information is stored in the directory */var/run/faillock/*. The locking information for an account is stored in a unique file (named after the account name) under this directory. Using the *pam_faillock.so* application which modifies this file, the account can be unlocked. The DAC permissions of that file ensure that only the root user can write to the file.

6.1.54 FMT_MTD.1(IAU) Management of TSF data

FMT_MTD.1.1 The TSF shall restrict the ability to **initialize, modify, delete** the **user security attributes *stored in local databases*** to [**assignment**:

a) **administrators,**

b) **users authorized to modify their own general information**

]

Application Note: This SFR applies to FIA_ATD.1, FIA_UAU.1, FIA_UID.1.

Application Note: The configuration of these parameters is performed with the configuration

[3] “and matching of” is added compared with OSPP.

files */etc/passwd* and */etc/shadow* which are writeable to the root user only. The general information of a user is the GECOS field in file */etc/passwd*.

6.1.55 FMT_MTD.1(SSH) Management of TSF data

FMT_MTD.1.1 The TSF shall restrict the ability to **modify** the **authentication methods provided by the OpenSSH server** to **[assignment: administrators]**.

Application Note: This SFR applies to FIA_UAU.5.

Application Note: The configuration of this parameter is performed with the configuration file */etc/sshd_config*, which is writable to the root user only.

6.1.56 FMT_MTD.1(SSL) Management of TSF data

FMT_MTD.1.1 The TSF shall restrict the ability to **modify** the **time interval of user inactivity for locking an interactive session** to **[assignment:**

a) **administrators for system wide settings,**
b) **each user for his own sessions, if allowed by the root user.]**

Application Note: This SFR applies to FTA.SSL.1.

Application Note: A shell variable TMOUT is used to define the time interval before inactive session gets locked. The root user can set the variable in system-level configuration file (e.g. /etc/profile) globally, and each user can change it in his own session if the variable is NOT defined as ‘readonly’ by root.

6.1.57 FMT_MTD.1(AM-AP) Management of TSF data

FMT_MTD.1.1 The TSF shall restrict the ability to [**selection**: **modify, delete, clear]** the [**assignment**: **management of any TSF data]** to [**assignment**: **users allowed to invoke the application managing the TSF data or to edit the files holding the TSF data]** ***only after another user with the role [assignment: administrator] has approved the action***.

Application Note: The *sudo* tool allows the root user to specify which application is executed by what user with which UID. It allows the specification of the rules fine grained down to a single application for a single user with a single target UID, including root.

6.1.58 FMT_MTD.1(AM-MR) Management of TSF data

FMT_MTD.1.1 The TSF shall restrict the ability to **modify**, **[selection: delete**] the **assignment of roles to users down to the granularity of single users** to [**assignment**: **administrators**].

Application Note: The */etc/sudoers* file can be used to specify if each single user is granted to run some application as root, and the file is writable only to the root user.

6.1.59 FMT_MTD.1(AM-MD) Management of TSF data

FMT_MTD.1.1 The TSF shall restrict the ability to [**selection**: **delegate, revoke delegation of**] the [**assignment**: **administrator user role**] to **users granted that role**.

Application Note: The delegation is implemented using the *sudo* command. Every user that is allowed to use the root user can delegate parts of his responsibility by adding an appropriate rule into the */etc/sudoers* configuration file.

6.1.60 FMT_MTD.1(AM-MA) Management of TSF data

FMT_MTD.1.1 The TSF shall restrict the ability to **modify, [selection: delete, clear]** the **[assignment: approval of administrative actions**] to [**assignment**: **administrators**].

Application Note: The */etc/sudoers* file is accessible to the root user only based on the DAC permission bits.

6.1.61 FMT_REV.1(OBJ) Revocation

FMT_REV.1.1 The TSF shall restrict the ability to revoke **object security attributes defined by SFPs** associated with the **corresponding object** under the control of the TSF to [**assignment**:

a) **(for DAC permissions): owners of the object, and administrators;**
b) **(for Other security attributes): administrators.**

]

FMT_REV.1.2 The TSF shall enforce the ***following*** rules:

a) **The access rights associated with an object shall be enforced when an access check is made;**
b) **[assignment: no other revocation rules].**

6.1.62 FMT_REV.1(USR) Revocation

FMT_REV.1.1 The TSF shall restrict the ability to revoke **user security attributes defined by the SFP** associated with the **corresponding user** under the control of the TSF to [**assignment**: **administrators**].

FMT_REV.1.2 The TSF shall enforce the following rules:

a) **The enforcement of the revocation of security-relevant authorizations with the next user-subject binding process during the next authentication of the user;**
b) [**assignment**: **No other revocation rules**]

Application Note: The changes are enforced for **a new session** when the user affected by the change initiates that new session.

6.1.63 FMT_SMF.1 Specification of Management Functions

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

a) **Management of auditing;**
b) **Management of cryptographic network protocols;**
c) **Management of Persistent Storage Object Access Control Policy;**
d) **Management of Transient Storage Object Access Control Policy;**
e) **Management of Network Information Flow Control Policy;**
f) **Management of identification and authentication policy;**
g) **Management of user security attributes;**
h) **[assignment: no other management functions]**;

6.1.64 FMT_SMR.1 Security roles

FMT_SMR.1.1 The TSF shall maintain the roles:

a) **User role with the following rights:**
    i. **Users are authorized to modify their own user password;**
    ii. **Users are authorized to modify the access control permissions for the named objects they own;**
    iii. **[assignment: no other rights];**
b) **[assignment: administrator role with full privileges]**

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 above mentioned capabilities which, in the TOE configuration, is only provided to processes executing with 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 "wheel" group. This group allows the use of the "*su*" application which is the only way to assume the root user capabilities. In addition, the "*sudo*" application allows granting users the privilege to execute commands with a different user ID, including the root user.

Application Note: This SFR is hierarchical to the PP SFR of FMT_SMR.1 which satisfies the strict conformance claim.

6.1.65 FPT_STM.1 Reliable time stamps

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

6.1.66 FPT_TDC.1 Inter-TSF basic TSF data consistency

FPT_TDC.1.1 The TSF shall provide the capability to consistently interpret [**assignment**: **the following TSF data types**

a) **Packet filter: protocol headers for the network protocols covered by the packet filter;**

] when shared between the TSF and another trusted IT product.

FPT_TDC.1.2 The TSF shall use [**assignment**: **the following interpretation rules**

a) **Packet filter: protocol headers specification provided in RFC 791 (IP), RFC 793 (TCP), RFC 768 (UDP), RFC 792 (ICMP);**

] when interpreting the TSF data from another trusted IT product.

6.1.67 FTA_SSL.1 TSF-initiated session locking

FTA_SSL.1.1 The TSF shall lock an interactive session ***to a human user maintained by the TSF*** after [**assignment**: **an administrator-configurable time interval of user inactivity**] by:

a) clearing or overwriting ***TSF controlled*** display devices, making the current contents unreadable;

b) disabling any activity of the user's ***TSF controlled*** data access/***TSF controlled*** display devices other than unlocking the session.

FTA_SSL.1.2 The TSF shall require the following events to occur prior to unlocking the session:

a) **Successful re-authentication with the credentials of the user owning the session using [assignment: password based authentication];**

b) **[assignment: No other events**].

Application Note: It is possible that the TSF establishes a connection to a session on a remote trusted IT system, for example when using SSH. This remote trusted IT system maintains the session established with the communication channel. The locking requirement however applies to the session maintained by the TSF only as the TSF can only exercise control of the sessions it maintains.

6.1.68 FTA_SSL.2 User-initiated locking

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

a) clearing or overwriting ***TSF controlled*** display devices, making the current contents unreadable;

b) disabling any activity of the user's ***TSF controlled*** data access/***TSF controlled*** display devices other than unlocking the session.

FTA_SSL.2.2 The TSF shall require the following events to occur prior to unlocking the session:

a) **Successful re-authentication with the credentials of the user owning the session using [assignment: password based authentication];**

b) **[assignment: No other events to occur]**.

Application Note: It is possible that the TSF establishes a connection to a session on a remote trusted IT system, for example when using SSH. This remote trusted IT system maintains the session established with the communication channel. The locking requirement however applies to the session maintained by the TSF only, as the TSF can only exercise control of the sessions it maintains.

6.1.69 FTP_ITC.1 Inter-TSF trusted channel

FTP_ITC.1.1 The TSF shall provide a communication channel between itself and another 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~~ and*** disclosure ***using the following mechanisms:***

a) ***Cryptographically-protected communication channel using [assignment****:*
   - ***SSH protocol version 2 as defined in RFCs 4251, 4252, 4253, and 4254 with a combination of the following cipher suites defined there:***
     1) ***Symmetric ciphers defined in FCS_COP.1(NET) for encryption;***
     2) ***Keyed hash algorithms defined in FCS_COP.1(NET) for integrity;***
     3) ***Algorithms defined in FCS_CKM.2(NET) for key exchange;***
     4) ***Asymmetric ciphers defined in FCS_COP.1(NET) for public key encryption;***
   - ***TLS as defined in [RFC5246] using X.509 certificates and supporting the following cipher suites defined there:***
     1) ***Algorithms defined in FCS_COP.1(NET);***
   - ***IPSEC protocol ESP as defined in [RFC4303] using the cryptographic algorithms:***
     1) ***Symmetric ciphers defined in FCS_COP.1(NET) for ESP encryption;***
     2) ***Keyed hash algorithms defined in FCS_COP.1(NET) for ESP authentication and authentication header protection;***
     3) ***Hash algorithms defined in FCS_COP.1(NET) for key negotiation and SA establishment;***
     4) ***Algorithms defined in FCS_CKM.2(NET) for use in IKE key establishment;***
     5) ***Asymmetric ciphers defined in FCS_COP.1(NET) for Peer Authentication;***

   ]

b) ***[selection: no physically protected communication channel;***

c) ***[assignment: no other mechanisms for trusted***

communication channels]].

FTP_ITC.1.2 The TSF shall permit [**selection**: **the TSF, 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 and** [**assignment**: **no other conditions or functions**].

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.

6.1.70 FMT_MTD.1(TB) Management of TSF data

FMT_MTD.1.1 The TSF shall restrict the ability to **generate and update** the **integrity data base of the TSF and TSF data for the integrity verification of the TSF and TSF data** to **the administrator who can only operate in maintenance mode of the TOE*[4]***.

Application Note: The audit generation requirements specified by FAU_GEN.1 of the OSPP base do not apply to this SFR, as the integrity verification is performed without the remainder of the TSF, including the audit functionality, being active. If an integrity violation is detected, none of the TSF mechanisms providing a general-purpose operational environment to users is initialized, and users cannot interact with the TOE, its resources, TSF functions, TSF data, and user data.

6.1.71 FMT_SMR.2(TB) Restrictions on security roles

FMT_SMR.2.1 The TSF shall maintain the roles: **administrator who can only operate in maintenance mode of the TOE*[5]***.

FMT_SMR.2.2 The TSF shall be able to associate users with roles.

FMT_SMR.2.3 The TSF shall ensure that the conditions **of a maintenance mode where no user, administrator other than the administrator performing the integrity verification management, or external entity can interact with the TOE, resources controlled by the TOE, TSF functions, TSF data, or user data** are satisfied.

Application Note: The audit generation requirements specified by FAU_GEN.1 of the OSPP base do not apply to this SFR, as the integrity verification is performed without the remainder of the TSF, including the audit functionality, being active. If an integrity

[4] Root is the only administrator who can only operate in maintenance mode of the TOE.
[5] Root is the only who can only operate in maintenance mode of the TOE.

violation is detected, none of the TSF mechanisms providing a general-purpose operational environment to users is initialized, and users cannot interact with the TOE, its resources, TSF functions, TSF data, and user data.

6.1.72 FPT_TIM.1(TB) TSF integrity monitoring and action

FPT_TIM.1.1 The TSF ***hosted in a non-modifiable environment*** shall monitor **the TSF and TSF data loaded and executed before the TSF for integrity verification provided with the modifiable environment is active stored in containers controlled by the TSF** for **modification of that TSF and TSF data on all objects ~~*using the following rules:*~~ *based on the following attributes***:
[**selection**:

a) **cryptographic message digest of the TSF;**
b) **cryptographic message digest of the TSF data;**

]

FPT_TIM.1.2 Upon detection of a data integrity error, the TSF shall:

a) **Abort the entire TSF loading and execution process;**
b) **Inform the administrator about the violation;**
c) [**assignment**: **No additional action to be taken**]

Application Note: The audit generation requirements specified by FAU_GEN.1 of the OSPP base do not apply to this SFR, as the integrity verification is performed without the remainder of the TSF, including the audit functionality, being active. If an integrity violation is detected, none of the TSF mechanisms providing a general-purpose operational environment to users is initialized, and users cannot interact with the TOE, its resources, TSF functions, TSF data, and user data.

6.1.73 FDP_SDI.2(IV) Stored data integrity monitoring and action

FDP_SDI.2.1 The TSF shall ***be able to*** monitor user data stored in containers controlled by the TSF for **modification of that data** on all objects, based on the following attributes: [**selection**: **cryptographic keyed message digest of the user data**]

FDP_SDI.2.2 Upon detection of a data integrity error, the TSF shall **be able to perform one or more of the following actions**:

a) **Deny the subject's access request to the integrity-violated user data;**
b) **Inform the subject attempting to access the modified user data upon initial access;**
c) [**assignment**: **No additional action to be taken**].

Application Note: Integrity verification shall be performed at least at the time of initial access to the data. Data is initially accessed when the TSF establishes the link between the subject and the user data, or between the subject and the container holding the user data. For example, in UNIX-like environments, initial access occurs at the time of opening a file system object, the execution of a file,

or the initial access to IPC objects. Operations on these objects after the TSF has established the link (for example, reading or writing to the file system object or IPC object) are not considered to be the initial access.

6.1.74 FMT_MTD.1(IV-ACT) Management of TSF data

FMT_MTD.1.1 The TSF shall restrict the ability to **query, modify** [**selection**: **change_default**] the **actions to be performed upon detection of an integrity violation** to [**assignment**: **no user**].

Application Note: The action (i.e. “returning an error from Integrity Verification subsystem”) upon detection of integrity violation is hard-coded in the TOE, and no further action is specified for the user trying to open the protected file.

6.1.75 FMT_MTD.1(IV-TSF) Management of TSF data

FMT_MTD.1.1 The TSF shall restrict the ability to **enable, disable, select the TSF code or TSF data for, generate and update the integrity data base of the TSF data** for [**selection**: **query, modify**] the **integrity verification of the TSF code and TSF data** to [**assignment**: **administrators**].

6.1.76 FMT_MTD.1(IV-USR) Management of TSF data

FMT_MTD.1.1 The TSF shall restrict the ability to **enable, disable, select the user data for, generate and update the integrity data base of the user data** for [**selection**: **query, modify**] the **integrity verification of user data** to [**assignment**: **administrators**]].

6.1.77 FPT_TIM.1(IV) TSF integrity monitoring and action

FPT_TIM.1.1 The TSF shall ***be able to*** monitor **TSF code, TSF data** for **unauthorized modification of the TSF code and TSF data** using the following rules: [**selection**:

a) **verification of cryptographic keyed message digest of the TSF code;**
b) **verification of cryptographic keyed message digest of the TSF data;**

].

FPT_TIM.1.2 Upon detection of a data integrity error, the TSF shall **be able to perform one or more of the following actions**:

a) **Deny the subject's access request to the integrity-violated TSF code or TSF data;**
b) **Inform the administrator[6] about the integrity violation upon initial access;**
c) [**assignment**: **No additional action to be taken**]

[6] Root is the only administrator to get the violation information.

6.2 Rationale for Security Functional Requirements

This section provides the rationale for the internal consistency and completeness of the security functional requirements defined in this Protection Profile.

6.2.1 Security Requirements Coverage

SFR	Objectives
FAU_GEN.1	O.AUDITING
FAU_GEN.2	O.AUDITING
FAU_SAR.1	O.AUDITING
FAU_SAR.2	O.AUDITING
FAU_SEL.1	O.AUDITING
FAU_STG.1	O.AUDITING
FAU_STG.3	O.AUDITING
FAU_STG.4	O.AUDITING
FCS_CKM.1(SYM)	O.CRYPTO.NET
FCS_CKM.1(RSA)	O.CRYPTO.NET
FCS_CKM.1(DSA)	O.CRYPTO.NET
FCS_CKM.1(ECDSA)	O.CRYPTO.NET
FCS_CKM.2(NET)	O.CRYPTO.NET
FCS_CKM.4	O.CRYPTO.NET
FCS_COP.1(NET)	O.CRYPTO.NET
FCS_COP.1(CP)	O.CP.USERDATA
FCS_COP.1(IV)	O.INTEGRITY.TSF
FCS_RNG.1(SSL-DFLT)	O.CRYPTO.NET
FCS_RNG.1(DM-INIT)	O.CP.USERDATA
FCS_RNG.1(DM-RUN)	O.CP.USERDATA
FCS_RNG.1(NSS)	O.CRYPTO.NET
FCS_RNG.1(IV)	O.INTEGRITY.TSF O.INTEGRITY.USERDATA
FDP_ACC.1(PSO)	O.CP.USERDATA O.DISCRETIONARY.ACCESS
FDP_ACC.1(TSO)	O.CP.USERDATA O.SUBJECT.COM
FDP_ACF.1(PSO)	O.CP.USERDATA O.DISCRETIONARY.ACCESS
FDP_ACF.1(TSO)	O.CP.USERDATA O.SUBJECT.COM
FDP_IFC.2(NI)	O.NETWORK.FLOW

FDP_IFF.1(NI)	O.NETWORK.FLOW
FDP_ITC.2	O.DISCRETIONARY.ACCESS O.SUBJECT.COM O.NETWORK.FLOW
FDP_RIP.2	O.AUDITING O.CRYPTO.NET O.DISCRETIONARY.ACCESS O.SUBJECT.COM O.NETWORK.FLOW O.I&A
FDP_RIP.3	O.AUDITING O.CRYPTO.NET O.DISCRETIONARY.ACCESS O.SUBJECT.COM O.NETWORK.FLOW O.I&A
FIA_AFL.1	O.I&A
FIA_ATD.1(HU)	O.I&A
FIA_ATD.1(TU)	O.NETWORK.FLOW
FIA_SOS.1	O.I&A
FIA_UAU.1	O.I&A
FIA_UAU.5	O.I&A
FIA_UAU.7	O.I&A
FIA_UID.1	O.I&A O.NETWORK.FLOW
FIA_USB.2	O.I&A
FMT_MSA.1(PSO)	O.CP.USERDATA O.MANAGE
FMT_MSA.1(TSO)	O.CP.USERDATA O.MANAGE
FMT_MSA.3(PSO)	O.CP.USERDATA O.MANAGE
FMT_MSA.3(TSO)	O.CP.USERDATA O.MANAGE
FMT_MSA.3(NI)	O.MANAGE
FMT_MSA.4(PSO)	O.MANAGE
FMT_MTD.1(AE)	O.MANAGE
FMT_MTD.1(AS)	O.MANAGE
FMT_MTD.1(AT)	O.MANAGE
FMT_MTD.1(AF)	O.MANAGE
FMT_MTD.1(NI)	O.MANAGE
FMT_MTD.1(IAT)	O.MANAGE
FMT_MTD.1(IAF)	O.MANAGE
FMT_MTD.1(IAU)	O.MANAGE
FMT_MTD.1(SSH)	O.MANAGE

FMT_MTD.1(SSL)	O.MANAGE
FMT_MTD.1(AM-AP)	O.ROLE.APPROVE
FMT_MTD.1(AM-MR)	O.ROLE.MGMT
FMT_MTD.1(AM-MD)	O.ROLE.DELEGATE
FMT_MTD.1(AM-MA)	O.ROLE.APPROVE
FMT_REV.1(OBJ)	O.MANAGE
FMT_REV.1(USR)	O.MANAGE
FMT_SMF.1	O.MANAGE
FMT_SMR.1	O.MANAGE
FPT_STM.1	O.AUDITING
FPT_TDC.1	O.DISCRETIONARY.ACCESS O.SUBJECT.COM O.NETWORK.FLOW
FTA_SSL.1	O.I&A
FTA_SSL.2	O.I&A
FTP_ITC.1	O.TRUSTED_CHANNEL
FMT_MTD.1(TB)	O.INTEGRITY.BOOT.MGT
FMT_SMR.2(TB)	O.INTEGRITY.BOOT.MGT
FPT_TIM.1(TB)	O.INTEGRITY.BOOT
FDP_SDI.2(IV)	O.INTEGRITY.USERDATA O.INTEGRITY.ACTION
FMT_MTD.1(IV-ACT)	O.INTEGRITY.MANAGE
FMT_MTD.1(IV-TSF)	O.INTEGRITY.MANAGE
FMT_MTD.1(IV-USR)	O.INTEGRITY.MANAGE
FPT_TIM.1(IV)	O.INTEGRITY.TSF O.INTEGRITY.ACTION

Table 16 Security Functional Requirements coverage

6.2.2 Security Requirements Sufficiency

Objectives	Coverage Rationale
O.AUDITING	The events to be audited are defined in [FAU_GEN.1] and are associated with the identity of the user that caused the event [FAU_GEN.2]. Authorized users are provided the capability to read the audit records [FAU_SAR.1], while all other users are denied access to the audit records [FAU_SAR.2]. The authorized user must have the capability to specify which audit records are generated [FAU_SEL.1]. The TOE prevents the audit log from being modified or deleted [FAU_STG.1] and ensures that the audit log is not lost due to resource shortage [FAU_STG.3, FAU_STG.4]. To support auditing, the TOE is able to maintain proper time stamps [FPT_STM.1].

	The protection of reused resources ensures that no data leaks from other protected sources [FDP_RIP.2, FDP_RIP.3].
O.CRYPTO.NET	The cryptographically-protected network protocol [FCS_COP.1(NET)] is supported by the generation of symmetric keys [FCS_CKM.1(SYM)], as well as asymmetric keys [FCS_CKM.1(RSA), FCS_CKM.1(DSA), FCS_CKM.1(ECDSA)] , and the functionality is based on the random number generator as defined by [FCS_RNG.1(SSL-DFLT), FCS_RNG.1(NSS)]. As part of the cryptographic network protocol, the TOE securely exchanges the symmetric key with a remote trusted IT system [FCS_CKM.2(NET)]. The TOE ensures that all keys are zeroized upon de-allocation [FCS_CKM.4]. The protection of reused resources ensures that no data leaks from other protected sources [FDP_RIP.2, FDP_RIP.3].
O.DISCRETIONARY.ACC ESS	The TSF must control access to resources based on the identity of users that are allowed to specify which resources they want to access for storing their data. The access control policy must have a defined scope of control [FDP_ACC.1(PSO)]. The rules for the access control policy are defined [FDP_ACF.1(PSO)]. When import of user data is allowed, the TOE must ensure that user data security attributes required by the access control policy are correctly interpreted [FDP_ITC.2, FPT_TDC.1]. The protection of reused resources ensures that no data leaks from other protected sources [FDP_RIP.2, FDP_RIP.3].

O.NETWORK.FLOW	The network information flow control mechanism controls the information flowing between different entities [FDP_IFC.2(NI)]. The TOE implements a rule-set governing the information flow [FDP_IFF.1(NI)]. To facilitate the information flow control, the information must be identified [FIA_UID.1] based on security attributes the TOE can maintain [FIA_ATD.1(TU)]. The TOE must ensure that security attributes of the network data required by the information flow control policy are correctly interpreted [FDP_ITC.2, FPT_TDC.1]. The protection of reused resources ensures that no data leaks from other protected sources [FDP_RIP.2, FDP_RIP.3].
O.SUBJECT.COM	The TSF must control the exchange of data using transient storage objects between subjects based on the identity of users. The access control policy must have a defined scope of control [FDP_ACC.1(TSO)]. The rules for the access control policy are defined [FDP_ACF.1(TSO)]. When import of user data is allowed, the TOE must ensure that user data security attributes required by the access control policy are correctly interpreted [FDP_ITC.2, FPT_TDC.1]. The protection of reused resources ensures that no data leaks from other protected sources [FDP_RIP.2, FDP_RIP.3].
O.I&A	The TSF must ensure that only authorized users gain access to the TOE and its resources. Users authorized to access the TOE must use an identification and authentication process [FIA_UID.1, FIA_UAU.1]. Multiple I&A mechanisms are allowed as specified in [FIA_UAU.5]. To ensure authorized access to the TOE, authentication data is protected [FIA_ATD.1(HU), FIA_UAU.7]. Proper authorization for subjects acting on behalf of users is also ensured [FIA_USB.2]. The appropriate strength of the authentication mechanism is ensured [FIA_SOS.1]. To support the strength of authentication methods, the TOE is capable of identifying and reacting to unsuccessful authentication attempts [FIA_AFL.1]. In addition, user-initiated and TSF-initiated session locking [FTA_SSL.1, FTA_SSL.2] protect the authenticated user's session. The protection of reused resources ensures that no data leaks from other protected sources [FDP_RIP.2, FDP_RIP.3] are present.

O.MANAGE	The TOE provides management interfaces globally defined in [FMT_SMF.1] for: the access control policies [FMT_MSA.1(PSO), FMT_MSA.1(TSO), FMT_MSA.3(PSO), FMT_MSA.3(TSO)]; the information flow control policy [FMT_MSA.3(NI), FMT_MTD.1(NI)]; the auditing aspects [FMT_MTD.1(AE), FMT_MTD.1(AS), FMT_MTD.1(AT), FMT_MTD.1(AF)]; the identification and authentication aspects [FMT_MTD.1(IAT), FMT_MTD.1(IAF), FMT_MTD.1(IAU), FMT_MTD.1(SSH)]; the session locking threshold [FMT_MTD.1(SSL)]; Persistently stored user data is stored either in hierarchical or relational fashion, which implies an inheritance of security attributes from parent object [FMT_MSA.4(PSO)]. The rights management for the different management aspects is defined with [FMT_SMR.1]. The management interfaces for the revocation of user and object attributes is provided with [FMT_REV.1(OBJ) and FMT_REV.1(USR)].
O.TRUSTED_CHANNEL	The TOE provides a trusted channel protecting communication between a remote trusted IT system and itself [FTP_ITC.1].
O.INTEGRITY.BOOT.MGT	Management of the integrity verification mechanism is provided with [FMT_MTD.1(TB)]. In addition, a specific role for performing that management aspect is required by [FMT_SMR.2(TB)].
O.INTEGRITY.BOOT	The integrity verification mechanism of TSF code and TSF data loaded before the TSF integrity verification mechanism at runtime of the TOE is active is defined with [FPT_TIM.1(TB)].
O.INTEGRITY.TSF	The integrity verification mechanism for validating TSF data and TSF code is defined with [FCS_COP.1(IV), FPT_TIM.1(IV)].
O.INTEGRITY.USERDATA	The integrity verification mechanism for validating user data is defined with [FDP_SDI.2(IV)] and supported by FCS_RNG.1(IV).
O.INTEGRITY.ACTION	The TOE shall perform pre-defined actions upon detection of a breach of integrity; actions are defined by [FDP_SDI.2(IV)] for user data and [FPT_TIM.1(IV)] for TSF data and TSF code.
O.INTEGRITY.MANAGE	The management aspect for the integrity verification mechanism is covered by: • [FMT_MTD.1(IV-ACT)] covering management of

	actions performed by the TOE upon detection of integrity violation. • [FMT_MTD.1(IV-TSF)] covering update of the integrity verification database of the TSF data and selection of data subject to verification for TSF data. • [FMT_MTD.1(IV-USR)] covering update of the integrity verification database and selection of data subject to verification for user data.
O.ROLE.DELEGATE	The delegation of roles is defined and specified in [FMT_MTD.1(AM-MD)].
O.ROLE.MGMT	The definition and management of rights based on roles is defined in [FMT_MTD.1(AM-MR)].
O.ROLE.APPROVE	The approval mechanism for roles is defined with [FMT_MTD.1(AM-AP)], supported by management of the approval mechanism, i.e., specification of which roles can approve which operations [FMT_MTD.1(AM-MA)].
O.CP.USERDATA	The confidentiality protection mechanism for user data at rest is provided with the access control policy specified with [FDP_ACC.1(PSO)], [FDP_ACC.1(TSO)], [FDP_ACF.1(PSO)] and [FDP_ACF.1(TSO)], and supported by the cryptographic operations defined in [FCS_COP.1(CP)]. The confidentiality protection is implemented with cryptographic mechanisms where the symmetric keys are derived from a random number generator as defined by [FCS_RNG.1(DM-INIT), FCS_RNG.1(DM-RUN)]. The management of the confidentiality protection mechanism is covered by [FMT_MSA.1(PSO)], [FMT_MSA.1(TSO)], [FMT_MSA.3(PSO)] and [FMT_MSA.3(TSO)] covering the general management aspects.

Table 17 Security Functional Requirements rationale

6.2.3 Security Requirements Dependency Analysis

SFR	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.1
FAU_SAR.1	FAU_GEN.1	FAU_GEN.1
FAU_SAR.2	FAU_SAR.1	FAU_SAR.1
FAU_SEL.1	FAU_GEN.1	FAU_GEN.1
	FMT_MTD.1	FMT_MTD.1(AE)
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(SYM)	[FCS_CKM.2 or FCS_COP.1]	FCS_COP.1(NET)
	FCS_CKM.4	FCS_CKM.4
FCS_CKM.1(RSA)	[FCS_CKM.2 or FCS_COP.1]	FCS_COP.1(NET)
	FCS_CKM.4	FCS_CKM.4
FCS_CKM.1(DSA)	[FCS_CKM.2 or FCS_COP.1]	FCS_COP.1(NET)
	FCS_CKM.4	FCS_CKM.4
FCS_CKM.1(ECDSA)	[FCS_CKM.2 or FCS_COP.1]	FCS_COP.1(NET)
	FCS_CKM.4	FCS_CKM.4
FCS_CKM.2(NET)	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1]	FCS_CKM.1(SYM), FCS_CKM.1(RSA), FCS_CKM.1(DSA) FCS_CKM.1(ECDSA)
	FCS_CKM.4	FCS_CKM.4
FCS_CKM.4	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1]	FCS_CKM.1(SYM)
FCS_COP.1(NET)	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1]	FCS_CKM.1(SYM), FCS_CKM.1(RSA), FCS_CKM.1(DSA), FCS_CKM.1(ECDSA)
	FCS_CKM.4	FCS_CKM.4
FCS_COP.1(CP)	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1]	FCS_CKM.1(SYM), FCS_CKM.1(RSA), FCS_CKM.1(DSA), FCS_CKM.1(ECDSA)
	FCS_CKM.4	FCS_CKM.4
FCS_COP.1(IV)	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1]	FCS_CKM.1(SYM), FCS_CKM.1(RSA), FCS_CKM.1(DSA), FCS_CKM.1(ECDSA)
	FCS_CKM.4	FCS_CKM.4
FCS_RNG.1(SSL-DFLT)	No dependencies	
FCS_RNG.1(DM-INIT)	No dependencies	
FCS_RNG.1(DM-RUN)	No dependencies	
FCS_RNG.1(NSS)	No dependencies	
FCS_RNG.1(IV)	No dependencies	
FDP_ACC.1(PSO)	FDP_ACF.1	FDP_ACF.1(PSO)
FDP_ACC.1(TSO)	FDP_ACF.1	FDP_ACF.1(TSO)
FDP_ACF.1(PSO)	FDP_ACC.1	FDP_ACC.1(PSO)
	FMT_MSA.3	FMT_MSA.3(PSO)
FDP_ACF.1(TSO)	FDP_ACC.1	FDP_ACC.1(TSO)

	FMT_MSA.3	FMT_MSA.3(TSO)
FDP_IFC.2(NI)	FDP_IFF.1	FDP_IFF.1(NI)
FDP_IFF.1(NI)	FDP_IFC.1	FDP_IFC.2(NI)
	FMT_MSA.3	FMT_MSA.3(NI)
FDP_ITC.2	[FDP_ACC.1 or FDP_IFC.1]	FDP_ACC.1(PSO) FDP_ACC.1(TSO)
	[FTP_ITC.1 or FTP_TRP.1]	FTP_ITC.1
	FPT_TDC.1	FPT_TDC.1
		FDP_IFC.2(NI)
FDP_RIP.2	No dependencies	
FDP_RIP.3	No dependencies	
FIA_AFL.1	FIA_UAU.1	FIA_UAU.1
FIA_ATD.1(HU)	No dependencies	
FIA_ATD.1(TU)	No dependencies	
FIA_SOS.1	No dependencies	
FIA_UAU.1	FIA_UID.1	FIA_UID.1
FIA_UAU.5	No dependencies	
FIA_UAU.7	FIA_UAU.1	FIA_UAU.1
FIA_UID.1	No dependencies	
FIA_USB.2	FIA_ATD.1	FIA_ATD.1(HU)
FMT_MSA.1(PSO)	[FDP_ACC.1 or FDP_IFC.1]	FDP_ACC.1(PSO)
	FMT_SMR.1	FMT_SMR.1
	FMT_SMF.1	FMT_SMF.1
FMT_MSA.1(TSO)	[FDP_ACC.1 or FDP_IFC.1]	FDP_ACC.1(TSO)
	FMT_SMR.1	FMT_SMR.1
	FMT_SMF.1	FMT_SMF.1
FMT_MSA.3(PSO)	FMT_MSA.1	FMT_MSA.1(PSO)
	FMT_SMR.1	FMT_SMR.1
FMT_MSA.3(TSO)	FMT_MSA.1	FMT_MSA.1(TSO)
	FMT_SMR.1	FMT_SMR.1
FMT_MSA.3(NI)	FMT_MSA.1	NO, but satisfied with FMT_MTD.1(NI)
	FMT_SMR.1	FMT_SMR.1
FMT_MSA.4(PSO)	[FDP_ACC.1 or FDP_IFC.1]	FDP_ACC.1(PSO)
FMT_MTD.1(AE)	FMT_SMR.1	FMT_SMR.1
	FMT_SMF.1	FMT_SMF.1
FMT_MTD.1(AS)	FMT_SMR.1	FMT_SMR.1
	FMT_SMF.1	FMT_SMF.1
FMT_MTD.1(AT)	FMT_SMR.1	FMT_SMR.1
	FMT_SMF.1	FMT_SMF.1
FMT_MTD.1(AF)	FMT_SMR.1	FMT_SMR.1
	FMT_SMF.1	FMT_SMF.1
FMT_MTD.1(NI)	FMT_SMR.1	FMT_SMR.1
	FMT_SMF.1	FMT_SMF.1
FMT_MTD.1(IAT)	FMT_SMR.1	FMT_SMR.1
	FMT_SMF.1	FMT_SMF.1
FMT_MTD.1(IAF)	FMT_SMR.1	FMT_SMR.1
	FMT_SMF.1	FMT_SMF.1

FMT_MTD.1(IAU)	FMT_SMR.1	FMT_SMR.1
	FMT_SMF.1	FMT_SMF.1
FMT_MTD.1(SSH)	FMT_SMR.1	FMT_SMR.1
	FMT_SMF.1	FMT_SMF.1
FMT_MTD.1(SSL)	FMT_SMR.1	FMT_SMR.1
	FMT_SMF.1	FMT_SMF.1
FMT_MTD.1(AM-AP)	FMT_SMR.1	FMT_SMR.1
	FMT_SMF.1	FMT_SMF.1
FMT_MTD.1(AM-MR)	FMT_SMR.1	FMT_SMR.1
	FMT_SMF.1	FMT_SMF.1
FMT_MTD.1(AM-MD)	FMT_SMR.1	FMT_SMR.1
	FMT_SMF.1	FMT_SMF.1
FMT_MTD.1(AM-MA)	FMT_SMR.1	FMT_SMR.1
	FMT_SMF.1	FMT_SMF.1
FMT_REV.1(OBJ)	FMT_SMR.1	FMT_SMR.1
FMT_REV.1(USR)	FMT_SMR.1	FMT_SMR.1
FMT_SMF.1	No dependencies	
FMT_SMR.1	FIA_UID.1	FIA_UID.1
FPT_STM.1	No dependencies	
FPT_TDC.1	No dependencies	
FTA_SSL.1	FIA_UAU.1	FIA_UAU.1
FTA_SSL.2	FIA_UAU.1	FIA_UAU.1
FTP_ITC.1	No dependencies	
FMT_MTD.1(TB)	FMT_SMR.1	FMT_SMR.2(TB)
	FMT_SMF.1	FMT_SMF.1
FMT_SMR.2(TB)	FIA_UID.1	No, see the rationale below the table
FPT_TIM.1(TB)	No dependencies	
FDP_SDI.2(IV)	No dependencies	
FMT_MTD.1(IV-ACT)	FMT_SMR.1	FMT_SMR.1
	FMT_SMF.1	FMT_SMF.1
FMT_MTD.1(IV-TSF)	FMT_SMR.1	FMT_SMR.1
	FMT_SMF.1	FMT_SMF.1
FMT_MTD.1(IV-USR)	FMT_SMR.1	FMT_SMR.1
	FMT_SMF.1	FMT_SMF.1
FPT_TIM.1(IV)	No dependencies	

Table 18 Security Functional Requirements dependency analysis

Rationale for unresolved dependencies:

- FMT_MSA.3(NI): FMT_MTD.1(NI) is specified to require the management of security attributes for the Network Information Flow Control Policy, just as a potential FMT_MSA.1(NI) would have been specified. However, the Network Information Flow Control Policy is not required to be enforced when managing the security attributes, as the management aspect of the network information flow control functionality is not protected by the network information flow control mechanism. Therefore, FMT_MSA.1 is not applicable and is replaced with

FMT_MTD.1(NI).

- FMT_SMR.2(TB):The SFR is specified such that the TOE ensures that only the defined administrative role (and no other user) is active and can interact with the TOE in the maintenance mode. Hence, the administrative role is implicitly defined with the maintenance mode and does not depend on the identification of the user as specified with FIA_UID.1.

6.3 Security Assurance Requirements

The security assurance requirements for the TOE include components below as specified in [CC] part 3:

- Evaluation Assurance Level 4 components
- ALC_FLR.3

Assurance Class	Assurance Components
ADV: Development	ADV_ARC.1 Security architecture description ADV_FSP.4 Complete functional specification ADV_IMP.1 Implementation representation of the TSF ADV_TDS.3 Basic modular design
AGD: Guidance documents	AGD_OPE.1 Operational user guidance AGD_PRE.1 Preparative procedures
ALC: Life-cycle support	ALC_CMC.4 Production support, acceptance procedures and automation ALC_CMS.4 Problem tracking CM coverage ALC_DEL.1 Delivery procedures ALC_DVS.1 Identification of security measures ALC_LCD.1 Developer defined life-cycle model ALC_FLR.3 Systematic flaw remediation (augmented)
ASE: Security Target evaluation	ASE_CCL.1 Conformance claims ASE_ECD.1 Extended components definition ASE_INT.1 ST Introduction ASE_OBJ.2 Security objectives ASE_REQ.2 Derived security requirements ASE_SPD.1 Security problem definition ASE_TSS.1 TOE Summary Specification

ATE: Test	ATE_COV.2 Analysis of coverage ATE_DPT.1 Testing: basic design ATE_FUN.1 Functional Testing ATE_IND.2 Independent Testing - sample
AVA: Vulnerability assessment	AVA_VAN.3 Focused Vulnerability analysis

No operations are applied to the assurance components apart from the operation to ASE_CCL.1 as defined in [OSPP].

6.3.1 ASE_CCL.1 refinement

ASE_CCL.1 specified in CC Part 3 is refined as follows: All Developer Action Elements, Content and Presentation Elements, Evaluator Action Elements remain unaltered, except the following:

ASE_CCL.1.10C The conformance claim rationale shall demonstrate that the statement of security requirements is consistent with the statement of security requirements in the PPs ***including the statements marked as "ST- Author Note" and the specification given in section 8.1 of the OSPP base*** for which conformance is being claimed.

6.4 Rationale for Security Assurance Requirements

The rationale for the refinement of ASE_CCL.1 is provided in [OSPP].

The basis for the justification of EAL4 requirements augmented with ALC_FLR.3 is the threat environment experienced by the typical consumers of the TOE. This matches the package description for EAL4 (enhanced-basic).

7 TOE Summary Specification

This section explains how the security functions are implemented. The different TOE security functions cover the various SFR classes.

7.1 Audit

The Lightweight Audit Framework (LAF) is used in the audit subsystem of the TOE, which is compliant with the requirements from Common Criteria. The TOE kernel implements the core of the LAF functionality. It gathers all audit events, analyzes these events based on the audit rules, collects related information, and forwards the audit events that are requested to be audited to the audit daemon (*auditd*) executing in user space.

The audit functionality of the OS kernel is controlled by an audit management tool (*auditctl*(8)) in user space, which communicates with the kernel through a specific *netlink* channel. This *netlink* channel is usable only by applications with the following capabilities:

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

The TOE Audit security functionality includes:

- Audit event selection
- Audit trail
- Audit log overflow protection
- Audit access protection

7.1.1 Audit event selection

LAF is able to intercept all system calls and retrieve audit log entries from privileged applications (or services) in user space. The audit subsystem allows selecting the events to be actually audited from the set of all possibly auditable events based on a group of audit rules. These audit rules are set in a rule configuration file (*/etc/audit/audit.rules*), and the kernel is then notified to build its own internal structure for the events to be audited.

The audit management tool (*auditctl*(8)) is used to load the audit rules from the rule configuration file. The audit rules can be modified at runtime of the system.

This security function covers the SFRs of: FAU_SEL.1.

7.1.2 Audit trail

An audit record consists of one or more lines of text in a format like “keyword=value”. The following information is contained in all audit record lines:

- Type: indicates the type of the event, such as SYSCALL, PATH, USER_LOGIN, or USER_MGMT;
- Timestamp: Date and time when the audit record was generated;
- Audit ID: unique numerical event identifier;
- Login ID (“auid”): the user ID of the user authenticated by the system (regardless if the user has changed his real and / or effective user ID afterwards);
- Effective user and group ID: the effective user and group ID of the process at the time the audit event was generated;
- Event outcom: Success or failure (where appropriate);
- Process ID of the subject that caused the event (PID);
- Hostname or terminal the subject used for performing the operation;
- Information about the intended operation;

Reliable time stamp is included in audit records. The information above is followed by event specific data. For example, for SYSCALL event records that involve file system objects, multiple text lines will be generated for a single event, all having the same time stamp and audit ID to permit easy correlation.

The audit trail is stored in ASCII text format. The TOE provides tools for managing audit trails, which can be used for post-processing of audit data. The tool, *ausearch*(8), allows selective extraction of records from the audit trail using defined selection criteria. It supports the specification of a fine-grained search pattern where each information component can be searched for, including combinations of these patterns.

This security function covers the SFRs of: FAU_GEN.1, FAU_GEN.2, FPT_STM.1, FAU_SAR.1.

7.1.3 Audit log overflow protection

When an event is to be audited according to the audit rules, the kernel would send it to the user space audit daemon (*auditd*) for storing, as an audit record, through the *netlink* channel mentioned above. The audit daemon writes the audit records to the audit trail. If the size of audit trail reaches a pre-configured warning threshold, the root user is notified about the condition and he can then backup audit trails and make room for new audit records. If the audit trail is full, the audit daemon would switch the system to single user mode or halt the system, depending on the setting in the configuration file *auditd.conf*.

This security function covers the SFRs of: FAU_STG.3, FAU_STG.4.

7.1.4 Audit access protection

Access to audit data (including the configuration files and audit trails) by normal users is prohibited by the discretionary access control function of the TOE. The permission is granted only to the root user.

This security function covers the SFRs of: FAU_GEN.1, FAU_GEN.2, FAU_SAR.2, FAU_SEL.1, FAU_STG.1, FAU_STG.3, FAU_STG.4, FMT_MTD.1(AE), FMT_MTD.1(AS), FMT_MTD.1(AT), FMT_MTD.1(AF).

7.2 Cryptographic services

The TOE offers different types of cryptographic services in different layers to protect user data:

- Cryptographic services in kernel, a socket interface is provided to user space applications to make use of this kind of the services; The package *cryptsetup* uses the crypto service in kernel space to manage dm-crypt and LUKS encrypted volumes.
- Cryptographic algorithms in several libraries for general use in user space. These libraries include *openssl* and *NSS*.
- Cryptographically secured network communication channels (**trusted channel**) to allow remote users to interact with the TOE. Using one type of the cryptographically secured network channels, a user can request the following services:
  - ➢ OpenSSH: The OpenSSH application suite provides access to the command line interface of the TOE. OpenSSH can provide interactive as well as non-interactive sessions, and the console provided via OpenSSH provides the same environment as a local console. OpenSSH implements the SSHv2 protocol. The cryptographic primitives are provided by OpenSSL;
  - ➢ Libreswan / Kernel: The Libreswan package implements the IKEv1 and IKEv2 protocols to securely establish the symmetric keys used for an IPSEC tunnel. These keys are handed to the kernel which implements the IPSEC protocol. The cryptographic primitives are provided by NSS.

7.2.1 SSHv2 Protocol

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

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

Reference	Description	Implementation Details
RFC 4253 chapter 5	Compatibility with old SSH versions	The OpenSSH implementation is capable of interoperating with clients and servers using the old 1.x protocol. That functionality is explicitly disabled in the evaluated configuration, it permits protocol version 2.0 exclusively.
RFC 4253 section 6.2	Compression	OpenSSH supports the OPTIONAL "zlib" compression method.

RFC 4253 section 6.3	Encryption	The ciphers supported in the evaluated configuration are listed in FCS_COP.1(NET) for the SSH protocol.
RFC 4252 chapter 7	Public Key Authentication Method: "publickey"	This REQUIRED authentication method is supported by OpenSSH but can be disabled by the administrator of the OpenSSH daemon.
RFC 4252 chapter 8	Password Authentication Method: "password"	This SHOULD authentication method is supported by OpenSSH but can be disabled by the administrator of the OpenSSH daemon.
RFC 4252 chapter 8	Password change request and setting new password	The OpenSSH implementation supports the optional password change mechanism in the evaluated configuration.
RFC 4252 chapter 9	Host-Based Authentication: "hostbased"	This OPTIONAL authentication method is disabled in the evaluated configuration.

Table 19 SSH implementation notes

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

This security function covers the SFRs of: FCS_CKM.1(RSA), FCS_CKM.1(ECDSA), FDP_RIP.2, FDP_RIP.3.

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 4.3 of [RFC4253] - the keys are generated using the random number generator of the underlying cryptographic library;
  - Diffie-Hellman key agreement as defined in section 6.1 of [RFC4253];
  - The keyed hash function for integrity protection as defined in section 4.4 of [RFC4253].

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

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

The OpenSSH applications of sshd, ssh and ssh-keygen use the OpenSSL random number generator seeded by pulling data from */dev/random* or */dev/urandom* to generate cryptographic keys. OpenSSL provides different DRNGs depending whether the FIPS 140-2 mode is enabled in the system.

This security function covers the SFRs of: FCS_CKM.1(SYM), FCS_CKM.2(NET), FCS_CKM.4, FCS_COP.1(NET), FCS_RNG.1(SSL-DFLT), FTP_ITC.1, FMT_SMF.1.

7.2.2 IPSEC and IKEv1 / IKEv2 Protocol Family

The TOE implements the protocol family of IPSEC and IKE. The Linux kernel handles the ESP / AH processing, and the *libreswan* package covers the IKE protocol suite processing as follows:

- Internet Key Exchange: The IKE protocol establishes the mutual session key used for encrypting the communication. Both endpoints that want to communicate via IPsec-protected channels must agree on a symmetric key that is used to encrypt data with. In fact, two keys are exchanged or agreed on, one for each communication direction for the IPSEC SA. In addition, as part of the IKE protocol, the key agreement for the ISAKMP SA is performed which protects the entire IKE communication. The IKE protocol is implemented by the *pluto* daemon and is solely provided with user space code.
- IPsec: Once the keys for the IPSEC SA are exchanged or agreed on, the encryption and decryption of the actual data that flows over the wire is covered with the IPsec protocols of ESP, potentially supported by AH. In the TOE, the kernel exclusively implements the IPsec protocol using the keys established with the IKE protocol.

The *pluto* IKE daemon part of Libreswan implements the IKEv1 and IKEv2 protocol. These protocols are specified in [RFC2409] and [RFC5996].

The IPsec implementation of the kernel supports the transport as well as the tunnel mode. This allows the configuration of a peer-to-peer, a peer-to-network or a network-to-network scenario.

The TOE supports the generation of the RSA, DSA, and ECDSA key pairs used by the client. The key generation mechanism uses the NSS random number generator. The evaluated configuration also allows the use of an externally-generated certificate. A widely accepted Certification Authority, or a server in a closed community acting as a CA, might be used to generate and/or sign such a certificate.

The following RFCs are supported for implementing the IPsec protocol family:

- [RFC2401], [RFC2402], [RFC2406], [RFC2407]: Defining of SPD/SAD, SA, AH, ESP;
- [RFC2408], [RFC2409]: ISAKMP, IKEv1;

- [RFC3526]: Diffie-Hellman groups;
- [RFC5114]: Diffie-Hellman groups;
- [RFC5996]: IKEv2;

This security function covers the SFRs of: FCS_CKM.1(SYM), FCS_CKM.1(RSA), FCS_CKM.1(DSA), FCS_CKM.1(ECDSA), FCS_CKM.2(NET), FCS_CKM.4, FCS_COP.1(NET), FCS_RNG.1(NSS), FTP_ITC.1, FMT_SMF.1, FDP_RIP.2, FDP_RIP.3.

7.2.3 TLS

The TOE provides TLSv1.1 and TLSv1.2 service. In contrast to the SSH protocol described above, the TLS protocol performs the support authentication as part by verifying the RSA certificates. The TOE can be configured using a bi-directional certificate verification where the client side validates the server certificate.

The following RFCs are supported for implementing the TLS protocol:

- [RFC4346]: TLS 1.1
- [RFC5246]: TLS 1.2

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

Reference	Description	Implementation Details
RFC 5246 section 7.3	Handshake protocol overview: certificates	The evaluated configuration always uses server certificates. Client certificates are used to allow the server to authenticate the client.
RFC 5246 appendix D.1	Random Number Generation and Seeding	OpenSSL uses data from the Linux kernel random number generator to seed the PRNG.
RFC 5246 appendix D.2	Certificates and authentication	The evaluated configuration supports verification of certificate chains.
RFC 5246 appendix D.3	Cipher suites	The ciphers supported in the evaluated configuration are listed in FCS_COP.1(NET) for the TLS protocol.
RFC 5246 appendix E	SSLv2, SSLv3, TLSv1.0, TLSv1.1 Backward Compatibility	The OpenSSL implementation supports the backwards compatible protocol, but this is disabled in the evaluated configuration. It permits use of TLSv1.1 TLSv1.2 exclusively.

Table 20 TLS implementation notes

The TOE supports the generation of the RSA, DSA, and ECDSA key pairs used by the client. The key generation mechanism uses the OpenSSL random

number generator. The evaluated configuration also allows the use of an externally-generated certificate. A widely accepted Certification Authority, or a server in a closed community acting as a CA, might be used to generate and/or sign such a certificate.

This security function covers the SFRs of: FCS_CKM.1(SYM), FCS_CKM.1(RSA), FCS_CKM.1(DSA), FCS_CKM.1(ECDSA), FCS_CKM.2(NET), FCS_CKM.4, FCS_COP.1(NET), FCS_RNG.1(SSL-DFLT), FTP_ITC.1, FDP_RIP.2, FDP_RIP.3.

7.2.4 Confidentiality protected data storage

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

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

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

For the cryptographic operation, the creator of the dm_crypt block device can select the cipher. When creating the dm_crypt block device, the master volume key is obtained from a random number generator and stored on the block device encrypted with the user's passphrase. The used random number generator depends on the state of the operating system at the time the master volume key is generated. The key derivation mechanism from the user's password is based on the LUKS mechanism which is also conformant to the FIPS140-2 cryptographic standard as it follows PBKDFv2.

The encryption and decryption operation of the block device is implemented by the kernel. To unlock the encrypted master volume key stored on the protected block device, the *cryptsetup(8)* application performs the following steps:

1) obtain the user's passphrase;
2) apply the LUKS key derivation mechanism on the passphrase;
3) read the encrypted master volume key from the block device;

4) decrypt the master volume key with the key derived from the user's passphrase;
5) inject the decrypted master volume key into the kernel;

Using the *cryptsetup(8)* tool, the master volume key can also be transferred by encrypting it with another passphrase which can be given to another user. The transfer follows the same steps outlined for the unlocking operation, but instead of injecting the decrypted session key into the kernel, *cryptsetup(8)* fetches the new passphrase from the user, applies the LUKS mechanism on that passphrase, encrypts the master volume key with the derived key and stores the encrypted session key in a separate area on the block device. At this point, the master volume key is now stored encrypted in two separate places.

Similarly, the *cryptsetup(8)* tool can also be used to erase the storage location of one encrypted master volume key, which implies that the user owning the passphrase of the affected encrypted session key is not able to unlock the block device any more.

During setup time of an encrypted disk, the application *cryptsetup(8)* uses data out of */dev/random* directly as key material (normal mode, runtime), */dev/urandom* directly as key material (normal mode, initial installation time), from the *libgcrypt* ANSI X9.31 DRNG seeded by */dev/random* (FIPS 140-2 mode, runtime), or from the *libgcrypt* ANSI X9.31 DRNG seeded by */dev/urandom* (FIPS 140-2 mode, initial installation time).

This security function covers the SFRs of FCS_COP.1(CP), FCS_RNG.1(DM-INIT), FCS_RNG.1 (DM-RUN), FDP_RIP.3.

7.3 Packet filter

The TOE kernel's network stack implementation follows the layering structure of the network protocols. It implements the code for handling the link layer as well as the network layer. For those layers, independent filter mechanisms are provided:

- Link layer: ebtables implements the filtering mechanism for bridges
- Network layer: netfilter/iptables implements the filtering mechanism for non-bridge interfaces

7.3.1 Network layer filtering

7.3.1.1 Netfilter

Netfilter is a framework for packet mangling, implemented in the Linux kernel network stack handling the network layer. The netfilter framework comprises of the following parts:

- The IP stack defines five hooks which are well-defined points in a network packet's traversal of the IP protocol stack. Each of the hooks, the network stack will call the netfilter framework allowing it to operate on the entire packet. Note: the netfilter framework provides such hooks in a number of

network protocol implementations, but the TOE only supports IP as outlined above. Therefore, the ST specification only covers the IP protocol.
- The netfilter framework provides register functions for other kernel parts to listen to the different hooks. When a packet traverses one of the hooks and passed to the netfilter framework, it invokes every registered kernel part. These kernel parts then can examine the packet and possibly alter it. As part of the examination, these kernel parts can instruct the netfilter framework to discard the packet, to allow it to pass, or to queue it to user space.
- When a packet is marked to be queued to user space, the netfilter framework handles the asynchronous communication with user space.

The netfilter framework implements the five hooks at the following points in the packet traversal chain:

- When the packet enters the network layer of the TOE and after applying some sanity checks, but before the routing table is consulted, the NF_IP_PRE_ROUTING hook is triggered.
- After passing the routing table decision and the routing code marks the packet to be targeted for another host, the NF_IP_FORWARD hook is triggered.
- After passing the routing table decision and the routing code marks the packet to be targeted for the local system, the NF_IP_LOCAL_IN hook is triggered.
- When the packet traversed the entire network stack and is about to be placed on the wire again, the NF_IP_POST_ROUTING hook is triggered.
- When a packet is generated locally, the NF_IP_LOCAL_OUT hook is triggered before the routing table is consulted.

7.3.1.2 Iptables

All communication on the network layer can be controlled by the *IPTables* framework. The TOE implements a packet filter as part of the network stack provided with the kernel. The combination of *IPTables* and *netfilter* implements the packet filter which provides stateful and stateless packet filtering for network communication by inspecting the IP header, the TCP header, UDP header and/or ICMP header of every network packet that passes the network stack.

The packet selection system called *IP Tables* uses the netfilter framework to implement the actual packet filtering logic on the network layer for the TCP/IP protocol family.

Note: IPTables is able to perform Network Address Translation (NAT) as well as Port Address Translation (PAT) for simple as well as more complex protocols. This mechanism is out of scope for the evaluation. Furthermore, packet mangling support is provided with IPTables which is also out of scope for the evaluation.

IPTables registers all hooks provided by the netfilter framework. The NAT/PAT mechanism uses the pre-routing and post-routing hooks whereas the packet filtering capability is enforced on the local-in, local-out and forwarding hooks.

IPTables consists of the following two components:

- In-kernel packet filter enforcement: The kernel-side of IPTables use the netfilter framework as indicated above. Three lists of packet filter rules are enforced by the kernel mechanism: one for each netfilter framework hook that applies to packet filtering. When a packet is analyzed by the IPTables kernel modules, they first select the applicable list based on the hook where the netfilter framework triggered IPTables. Each list contains zero or more rules which are iterated sequentially. A rule consists of a matching part (also called the "match extension") and an action part (also called the "target extension"). When a rule is applied to a packet, the kernel modules first apply the matching part of the rule. If the packet matches, the action part is enforced. If the action part contains a decision of the fate of the packet (to accept it, to drop it, or to drop it and sending a notification to the sender), the rule list validation stops for this packet. If the action part contains a modification instruction or log instruction for the packet, the rule list validation continues after performing this operation. When the rule list is iterated through and a packet could not be matched by a rule with a decision action (accept, drop), the default decision action applicable to the list is enforced. This default action is either to accept the packet, to drop the packet, or to drop the packet and send a notification to the sender.
- User space configuration application: The user space application *iptables(1)* allows the configuration of the IPTables kernel components. The application allows the specification of one rule per invocation where a rule contains the above mentioned matching part and action part. The tool also allows modification or deletion of existing rules as well as configuration of the default action. When using the tool, each invocation must specify the netfilter framework hook to which the rule applies to. See the man page of *iptables(1)* for more details.

This security function covers the SFRs of:

- Packet filtering rules: FDP_IFC.2(NI), FDP_IFF.1(NI);
- Interpretation of network protocol: FIA_UID.1, FDP_ITC.2, FPT_TDC.1;
- Maintenance of rules: FIA_ATD.1(TU), FMT_MTD.1(NI); FMT_MSA.3 (NI);

7.4 Identification and Authentication

User identification and authentication in the TOE includes all forms of interactive login (e.g. remote login using the SSH protocol or local login at the local console) as well as identity changes through the *su* and *sudo* commands. These all rely on explicit authentication information provided interactively by a user. In addition, the key-based authentication mechanism of the OpenSSH server is another form of authentication.

7.4.1 PAM-based identification and authentication mechanisms

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

- Providing login control and establishing all UIDs, GIDs and login ID for a subject;
- Ensuring the quality of passwords;
- Enforcing limits for accounts (such as the number of maximum concurrent sessions allowed for a user);
- Enforcing the change of passwords after a configured time including the password quality enforcement;
- Enforcement of locking of accounts after failed login attempts;
- Restriction of the use of the root account to certain terminals;
- Restriction of the use of the *su* and *sudo* commands

The login processing sets the following attributes for the user at login time:
- the real, effective, file system UID;
- the real, effective, file system GID;
- the set of supplementary GIDs of the subject that is created;
- the login UID;

Apparently, it is up to the client application, which is usually provided by a remote system, to protect the user’s entry of a password correctly (e. g. provide only obscured feedback).

During login processing, the user is shown a banner. After successful authentication, the login time is recorded.

When configuring the OpenSSH server, the administrator is allowed to enable SSH key-based authentication in addition or instead of the username/password based authentication. If both are enabled, the key-based authentication has the priority. When a user can successfully authenticate using the SSH key-based authentication based on a private SSH key in his possession, the TOE grants the user access.

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 either automatically after a configurable period of inactivity for that session, or upon the user's request.

This security function covers the SFRs of FDP_RIP.3, FIA_AFL.1, FIA_SOS.1, FIA_UAU.1, FIA_UID.1, FIA_UAU.5, FIA_UAU.7, FIA_USB.2, FMT_MTD.1(IAT), FMT_MTD.1(IAF), FMT_MTD.1(IAU).

7.4.2 User Identity changing

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

7.4.2.1 su command

The *su* command is intended for a switch to another identity that establishes a new login session and spawns a new shell with the new identity. When invoking *su*, the user must provide the credentials associated with the target identity, i.e., when the user wants to switch to another user ID, it has to provide the password protecting the account of the target user.

The primary use of the *su* command within the TOE is to allow appropriately authorized individuals the ability to assume the root identity to perform administrative actions. In this system the capability to login as the root identity has been restricted to defined terminals only. In addition, the use of the *su* command to switch to root has been restricted to users belonging to a special group (*wheel*). Users that don't have access to a terminal where root login is allowed and are not member of that special group will not be able to switch their real, effective, and file system user ID to root even if they would know the authentication information for root. 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.

7.4.2.2 sudo command

The *sudo* command is intended for giving users permissions to execute commands with another user identity. When invoking *sudo*, the user has to authenticate with his own credentials. *sudo* is associated with sophisticated ruleset that can be engaged to specify:

- source user ID;
- originating from which host;
- with which new user identity;
- can access a list of commands, possibly with specific configuration flags, or all commands within a directory.

7.4.2.3 Changed identities

When switching identities using *su* or *sudo*, 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).

Note: The login ID is not retained for the following special case:

1) User A logs into the system.
2) User A uses *su* to change to user B.
3) User B now edits the *cron* or *at* job queue to add new jobs. This operation is appropriately audited with the proper login ID.
4) Now when the new jobs are executed as user B, the system does not provide the audit information that the jobs are created by user A.

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

This security function covers the SFRs of FIA_USB.2, FMT_MTD.1(AM-AP), FMT_MTD.1(AM-MR), FMT_MTD.1(AM-MD), FMT_MTD.1(AM-MA), FMT_REV.1 (USR).

7.4.3 Authentication Data Management

Each TOE instance maintains its own set of users with their passwords and attributes. Although the same human user may have accounts on different servers interconnected by a network and running an instantiation of the TOE, those accounts and their parameter are not synchronized on different TOE instances. As a result, the same user may have different user names, different user Ids, different passwords and different attributes on different machines within the networked environment. Existing mechanism for synchronizing this within the whole networked system are not subject to this evaluation.

Each TOE instance within the network maintains its own administrative database by making all administrative changes on the local TOE instance. System administration has to ensure that all machines within the network are configured in accordance with the requirements defined in this Security Target.

The file */etc/passwd* contains following information for each user:

- the user's name;
- the id (an integer) of the user;
- an indicator whether the password of the user is valid;
- the principal group id (an integer) of the user;
- and other (not security relevant) information.

The file */etc/shadow* contains following information for each user:

- the user's name;
- a hash of the user's password ('*' for disabled account);
- the time the password was last changed;
- the expiration time of the password ('0' for changing at any time);
- the validity period of the password;
- time to warn user of an expiring password (7 for a full week);
- and some other information that are not subject to the security functions as defined in this Security Target.

Users are allowed to change their passwords by using the *passwd(1)* command. This application is able to read and modify the contents of file */etc/shadow* for the user's password entry, which would ordinarily be inaccessible to a non-privileged user process. Users are also warned to change their passwords at login time if the password will expire soon, and are prevented from logging in if the password has expired.

The time of the last successful logins is recorded in the directory */var/run/faillock/*, where one file per user is kept.

The TOE displays informative banners before or during the login to users. The banners can be specified in file */etc/issue* for logins via the physical console, or in file */etc/issue.net* for remote logins, such as via SSH. When performing a login on the physical console, the banner is displayed above the username and

password prompt. For logins via SSH, the banner is displayed to the remote peer during the SSH-session handshake takes place. The remote SSH client will display the banner to the user. When using the provided OpenSSH client, the banner is displayed when the user instructs the OpenSSH client to log into the remote system.

This security function covers the SFRs of FIA_ATD.1(HU).

7.4.4 SSH key-based authentication

In addition to the PAM-based authentication outlined above, the OpenSSH server is able to perform a 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 key-based authentication, a user first has to generate an RSA or ECDSA key pair. The private part of the key pair remains on the client side. The public part is copied to the server into the file *.ssh/authorized_keys* which resides in the home directory of the user he wants to log on as. When the login operation is performed, the SSHv2 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 process for the publickey authentication is defined in [RFC4252] 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. See *ssh-keygen(1)* on the TOE for more information.

This security function covers the SFRs of FIA_UAU.1, FIA_UID.1, FIA_UAU.5, FIA_SOS.1, FMT_MTD.1(SSH), FMT_MTD.1(SSL).

7.4.5 Session locking

The TOE uses the *screen(1)* application which locks the current session of the user either after an administrator-specified time of inactivity or upon the user's request.

To unlock the session, the user must supply his password. Screen uses PAM to validate the password and allows the user to access his session after a successful validation.

This security function covers the SFRs of FTA_SSL.1, FTA_SSL.2.

7.5 Discretionary Access control

Several discretionary access control (DAC) mechanisms are implemented in EulerOS, however, only the standard DAC mechanism in traditional Unix systems, i.e. permission-bits, is covered in this evaluation. The general policy

for DAC 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 may possess one or more of the following capabilities which provide the following exemptions from the DAC mechanism:

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

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 (i.e. file system objects and IPC objects). DAC is implemented with permission bits.

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 that 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);
- Process signals.

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

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

7.5.1 Permission bits

The DAC mechanism covered in this evaluation is the standard UNIX permission bits for file system objects in all supported file systems. There are three sets of three-bit tuple that define access for three categories of users: 1) the owning user, 2) users in the owning group, and 3) other users. The three-bit tuple in each set indicate the access permissions granted to each user category: one bit for read (r), one for write (w) and one for execute (x). Note that write access to file systems mounted as read only (e. g. CD-ROM) is always rejected (the exceptions are character and block device files which can still be written to as write operations do not modify the information on the storage media). Also, write access to file system objects marked as immutable is always rejected. The “restricted deletion” (or sticky) attribute is used for world-writable temp directories, preventing the removal of files by users other than the owner.

Each process has an inheritable “umask” attribute which is used to determine the default access permissions for new objects. It is a bit mask of the user/group/other read/write/execute bits, and specifies the access bits to be removed from new objects. For example, setting the umask to “002” ensures that new objects will be writable by the owner and group, but not by others. The umask is defined by the root user in the */etc/login.defs* file or “022” by default if not specified.

This security function covers the SFRs of FDP_ACC.1(PSO), FDP_ACF.1(PSO), FDP_ITC.2, FDP_RIP.2.

7.5.2 File system objects

Access to file system objects is generally governed by permission bits. File system objects access checks are performed when the object is initially opened, and are not checked on each subsequent access. Changes to access controls (i.e., revocation) are effective with the next attempt to open the object.

7.5.3 IPC objects

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.

Two other types of IPC objects, UNIX domain socket special files and Named Pipes, are represented as file system objects, and, the access control mechanism covering file system objects are applicable to these IPC mechanisms too.

The TOE maintains another IPC object type where each process has its own namespace for that: 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.

Modification of DAC attributes is restricted to the owner of the object or users with the aforementioned capabilities.

This security function covers the SFRs of FDP_ACC.1(TSO), FDP_ACF.1(TSO), FMT_REV.1(OBJ).

7.6 Security Management

The security management facilities provided by the TOE are usable by authorized users and/or authorized administrators to modify the configurations of TSF. The configurations of TSF are 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 users as well as 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 where usually the root user only is allowed to write to the files. In some special cases (like for /etc/shadow), the file is even readable to the root user only. It is the task of the persons responsible for setting up and administrating the system 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 administrative user 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 users, if applicable, by either using DAC (for virtual file system objects) or special kernel-internal verification checks for each interface.

The TOE provides security management applications for all security-relevant settings listed throughout this ST, i.e., all FMT_MSA.1, FMT_MSA.3, FMT_MSA.4, and FMT_MTD.1(*) iterations. Also FMT_SMF.1

7.6.1 Privileges

The TOE maintains two roles, the administrator and normal users. By default, only the root user is in the administrator role, and can perform any operations on the system. Normal users can only operate on objects they own or they are granted by the administrator to operate on. The administrator can also delegate his privilege to specified normal users after successful authentication.

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

Functionality accessible in user space are applications that can be invoked by users. Also, data accessible in user space is either data maintained with an application or data stored in persistent or transient storage objects. Privileges are controlled by permissions to invoke applications and to access data. For example, the configuration files including the user databases of */etc/passwd* and */etc/shadow* are accessible to the root user only. Therefore, the root user is given the privilege to perform modifications on this configuration data, which constitutes administrative actions.

Functionality and data maintained by the kernel must be accessed using system calls. The kernel implements a privilege check for functions and data that shall not be accessible by normal users. These privileges are controlled with capabilities that can be assigned to processes. If a process is assigned with a capability, it is allowed to request special operations that other processes cannot. To implement consistency with the Unix legacy, processes with the effective UID of zero are implicitly given all capabilities. However, these processes may decide to drop capabilities. Such capabilities are marked by names with the prefix of "CAP_" throughout this document. The Linux kernel implements many more capabilities than mentioned in this document. These unmentioned capabilities protect functions that do not directly cover SFR functionality but need to be protected to ensure the integrity of the system and its resources.

7.6.2 Approval and delegation of management functions

Using the *sudo* command, authorized administrators can approve that other users can perform management tasks. Once the administrator approves the operation, the */etc/sudoers* file is modified to grant the user the right to perform the administrative operation.

Using the */etc/sudoers* file, the administrator can specify the approval rules based on the following fine-grained properties:

- Specification of the command that can be executed. The command may contain wild cards;
- Specification of the target user ID or group ID the command shall be executed with (or “*RunAs*”);
- Specification of the user ID or group ID (where all members of the group are covered) which are allowed by this rule.

Using the *sudo* command and the associated */etc/sudoers* configuration file, the administrative users, i.e. the users allowed to use the root UID, are allowed to delegate parts or all of their authority to other users.

This security function covers all SFRs of FMT_MTD.1(AM-AP), FMT_MTD.1(AM-MR), FMT_MTD.1(AM-MD), FMT_MTD.1(AM-MA) , FMT_SMR.1, FMT_REV.1(OBJ), FMT_REV.1(USR).

7.7 Trusted boot and integrity verification

Integrity of kernel image, kernel modules and system executables is verified using the combination of **Secure Boot** and **IMA-appraisal** mechanisms. No special hardware is needed for Secure Boot; however, TPM is required to do IMA-appraisal. The trust chain at boot time looks like this:

CRTM -> UEFI -> shim -> GRUB2 -> OS kernel -> Application

The first two parts, CRTM and UEFI, are for hardware and firmware, and are out of the scope of this ST. One or more platform certificates (called PKs) are integrated into firmware, which are the root of the trust chain above. There are several booting stages covering the other parts of the trust chain for system and code integrity check at boot time and at run time.

7.7.1 Secure Boot for booting integrity

Secure boot contains the following steps.

- Pre-booting. *Shim* is the first-stage of boot loader, containing a self-signed CA certificate. It is signed by one of the PKs. At this stage, *shim* is loaded and verified by the firmware using one PK. If the verification fails, an error message will be prompted and the boot process gets blocked; otherwise, *shim* will be launched, and try to load GRUB2. Actually this pre-booting is out of the scope of this ST.
- Booting stage 2. Once verified and run by firmware, *shim* tries to load GRUB2. After loading, *shim* begins to verify GRUB2 using the self-signed CA embedded in *shim*. If the verification of GRUB2 failed, an error will be reported and the boot process will terminate; otherwise, *shim* will launch GRUB2 and the boot process continues.
- Booting stage 3. If launched, GRUB2 will load OS kernel and try to verify it. Generally, OS kernel is signed by the vendor using the same key as the one for signing GRUB2. GRUB2 calls back into *shim* to verify the signature of the kernel. If the signature cannot be verified, an error will be shown, and the booting process will be blocked; otherwise, the kernel will be started.
- Loading kernel modules. There is another certificate integrated in OS kernel, which is generated with RSA and key size 2048. All kernel modules are signed by the OS vendor using the private key paired with the certificate, and will always be verified using the certificate before being loaded into kernel. Furthermore, the TSF includes a Code Integrity Verification mechanism, whereby kernel modules will be loaded only if they are digitally signed by either Huawei or a trusted root certificate authority recognized by Huawei. This mechanism uses public-key

cryptography technology to verify the digital signature of each kernel module when it is to be loaded.

When trying to load a kernel module, the TSF compares two hash values of the module: one is the decrypted version of the value included within the module file using the public key stored in the certificate; the other is computed based on the code in the kernel module using cryptographic libraries in the TSF. If the two hash values match, the kernel module can then be loaded as usual. If all modules required are verified successfully by kernel at boot time, the secure boot process can terminate normally, and the first user process can start then. If the verification fails, the kernel will report the error message and will stop the booting process.

It is worth noting that module verification is always done when loading a kernel module, both at boot time and at normal runtime of the system. If a module fails in verification at normal runtime, the load command would abort with an error message, and the kernel would NOT be impacted.

If all modules are verified successfully by kernel at boot time, the secure boot process can terminate normally. After that, file integrity verification is transferred to IMA-appraisal.

7.7.2 IMA-appraisal for code integrity

IMA-appraisal is an integrity component in Linux kernel, cooperating with EVM, another protection mechanism in kernel. An IMA-appraisal policy file can be customized through an interface by the system administrator to specify a runtime measurement list. At system runtime, the files which match the rules in the policy will be protected.

EVM is used to detect offline tampering of the extended attributes in *security* class (e.g. *security.selinux*, *security.SMACK64*, *security.ima*). At the initialization stage, HMAC-sha256 of a set of security related extended attributes is calculated and stored in extended attribute *security.evm*. At runtime, the integrity of the security related extended attributes is verified in kernel by re-calculating their digest using HMAC-SHA256 and comparing the result with the value stored in their extended attribute *security.evm*. If the verification fails, this means that either the *security.evm* or other security related extended attributes are tampered.

There are several stages in protecting system integrity using IMA-appraisal.

- Base lining. If it can be ensured that the system enters a *secure* state after boot (e.g. when this is the first boot after installation), the system administrator can create the base line for files to be protected. This is done in FIX mode of the system. A few steps are needed to do the base lining:

  1) Enter FIX mode. Add kernel options ‘ima_appraise=fix’ and ‘evm=fix’ in file *grub.cfg*, and reboot the operating system. Then the system enters into FIX mode. A hash function used to generate extended attribute

security.ima of files can also be specified in a kernel option. It is default to SHA256.

2) Define the IMA-appraisal policy file (*/etc/ima/ima-policy*) and start IMA-appraisal. The policy file can be customized by system administrator. IMA-appraisal is started automatically at boot time if the policy file exists. It can also be started by writing the policy to file */sys/kernel/security/ima/policy*.
3) Generate two keys using *keyctl* command: **trusted key**, and evm encryption key (**evm-key**). The evm-key is used when generating and verifying extended attribute *security.evm* of files and is protected by the trusted key. Both the **trusted key** and the **evm-key** are generated by TPM.
4) Export the trusted key to non-volatile storage. The exported trusted key is protected by TPM after sealing;
5) Export evm-key to non-volatile storage. The exported evm-key is protected by trusted key based on AES256-CBC algorithm;
6) Enable EVM functionality before labeling files to be protected. This is done by writing '1' to file */sys/kernel/security/evm*.
7) Labeling files. This is to initialize the extended attributes used by IMA and EVM for files to be protected according to the IMA-appraisal policy. In the FIX mode, access these files and then their extended attribute *security.ima* will be generated automatically by kernel. As EVM is already enabled in the previous step, the extended attribute *security.evm* will be also generated accordingly.

- Verifying. This is done at normal runtime of the system.

1) Enter normal runtime. To do this, remove the kernel options 'ima_appraise=fix' and 'evm=fix' in file *grub.cfg*, and reboot the operating system.
2) Start IMA-appraisal. This is done automatically by kernel at boot time if the file */etc/ima/ima-policy* exist. It can also be done by writing the policy to file */sys/kernel/security/ima/policy*.
3) Load keys. Load trusted key and evm-key from non-volatile storage into system keyring using the command *evmctl*.
4) Enable EVM. This is done by writing '1' to file */sys/kernel/security/evm*. After this, the system enters normal runtime with IMA functionality.
5) Access files as usual. If a file is accessed that is protected according to the IMA-appraisal policy, the kernel will check its extended attributes *security.evm* and *security.ima*, and then verify the integrity of the file content. If any step failed, the file is regarded to be tampered and the access is denied.

This security function covers all SFRs of FCS_CKM.1(SYM), FCS_COP.1(IV), FCS_RNG.1(IV), FMT_MTD.1(TB), FMT_SMR.2(TB), FPT_TIM.1(TB), FDP_SDI.2(IV), FMT_MTD.1(IV-ACT), FMT_MTD.1(IV-TSF), FMT_MTD.1(IV-USR), FMT_TIM.1(IV).

8 Protection Profile Conformance Claim

This section provides the protection profile and extended package conformance claim and supporting justifications and rationale.

8.1 Rationale for Conformance to Protection Profile and Extended Package

This Security Target is in compliance with the Operating System Protection Profile, BSI-CC-PP-0067, version 2.0. and the following extended packages:

- Extended Package - Trusted Boot, version 2.0
- Extended Package - Integrity Verification, version 2.0
- Extended Package - Advanced Management, version 2.0

The TOE type defined in this ST (section 1.3.2) is a general purpose operating system, which is strictly compliant with the TOE type defined in the Protection Profile section 1.2.1.

The security problem definition and the security objectives have been exactly copied from the Protection Profile (section 5 and 6) and considering the modifications defined within the claimed extended packages.

All the security requirements defined in the Protection Profile and the claimed extended packages have been included in this ST (section 6) including all the Application Notes and statements marked as “ST-Author Note”.

9 Abbreviations

Abbreviation	Description
AH	Authentication Header
CC	Common Criteria
DAC	Discretionary Access Control
EAL	Evaluation Assurance Level
ESP	Encapsulating Security Payload
EVM	Extended Verification Module
HMAC	Hash-based Message Authentication Code
IKE	Internet Key Exchange
IMA	Integrity Measurement Architecture
IPA	Identity, Policy and Audit. An identity management system.
IPC	Inter Process Communication
IPSEC	IP Security Protocol
IV	Initial Vector (for crypto operations) Integrity Verification (for system integrity protection)
LUKS	Linux Unified Key Setup
MAC	Mandatory Access Control
NSS	Network Security Services
OSPP	Operating System Protection Profile
PBKDF2	Password-Based Key Derivation Function 2
PP	Protection Profile
RNG	Random number generator. Cryptographically secure pseudorandom number generator (CSPRNG) Deterministic random number generator (DRNG). Non-physical true random number generators (NPTRNG). Pseudo-RNG (PRNG) Physical true RNG (PTRNG). True random-number generator (TRNG).
SA	Security Association (in IPSEC or IKE)
SAR	Security Assurance Requirement
SFP	Security Function Policy
SFR	Security Functional Requirement
SSH	Secure Shell
ST	Security Target
TLS	Transport Layer Security
TOE	Target of Evaluation
TSF	TOE security function
TSFI	TSF Interface
TSP	TOE security policy