Red Hat Enterprise Linux, Version 7.1

Version:	**0.21**
Status:	**Released**
Last Update:	**2016-06-09**
Classification:	**public**

Trademarks

Red Hat and the Red Hat logo are trademarks or registered trademarks of Red Hat, Inc. in the United States, other countries, or both.

atsec is a trademark of atsec information security GmbH

Linux is a registered trademark of Linus Torvalds.

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

IBM, IBM logo, bladecenter, eServer, iSeries, OS/400, , POWER3, POWER4, POWER4+, pSeries, System p, POWER5, POWER5+, POWER6, POWER6+, POWER7, POWER7+, System x, System z, S390, xSeries, zSeries, zArchitecture, and z/VM are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

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

This document is based in parts on the Red Hat Enterprise Linux Version 6.2 Security Target, Copyright © 2013 by Red Hat, Inc. and atsec information security corp.

Legal Notice

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

This document may be reproduced or distributed in any form without prior permission provided the copyright notice is retained on all copies. Modified versions of this document may be freely distributed provided that they are clearly identified as such, and this copyright is included intact.

Revision History

Revision	Date	Author(s)	Changes to Previous Revision
0.210	2016-06-09	Stephan Mueller	ST for RHEL 7.1 derived from RHEL 6.2 ST

Table of Contents

List of Tables

1 Introduction

1.1 Security Target Identification

Title:	Red Hat Enterprise Linux, Version 7.1
Version:	0.21
Status:	Released
Date:	2016-06-09
Sponsor:	Red Hat, Inc.
Developer:	Red Hat, Inc.
Certification Body:	BSI
Certification ID:	BSI-DSZ-CC-0999
Keywords:	Security Target, Common Criteria, Linux Distribution, Embedded Linux

1.2 TOE Identification

The TOE is Red Hat Enterprise Linux Version 7.1.

1.3 TOE Type

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

1.4 TOE Overview

1.4.1 Configurations defined with this ST

This security target documents the security characteristics of the Red Hat Enterprise Linux distribution (abbreviated with RHEL throughout this document).

1.4.2 Overview description

Red Hat Enterprise Linux is a highly-configurable Linux-based operating system which has been developed to provide a good level of security as required in commercial environments. It also meets all requirements of the Operating System protection profile [OSPP]. Additional functionality to the OSPP base is claimed:

- Advanced Management (MLS mode only)
- Disk encryption
- Labeled Security (MLS mode only)
- Runtime protection against programming errors
- Linux Container

The TOE can operate in two different modes of operation called “Base mode” and “MLS mode”. In Base mode the SELinux security module does not enforce a mandatory access control policy for the general computing environment and does not recognize sensitivity labels of subjects and objects. SELinux can either be disabled completely, or enabled with a non-MLS policy which only add

additional restrictions to the base access control functions without interfering with the “root” administrator role. In this mode the TOE enforces all security requirements of the [OSPP]. SELinux must be enabled if the administrator wants to provide virtual machines.

In MLS mode the SELinux security module is configured to enforce the mandatory access control policy based on the labels of subjects and objects as required by the extended package for labeled security as well as advanced management. Note that a system in MLS mode can optionally be configured to use a single sensitivity label for all subjects and objects to provide an operational mode equivalent to pure role-based access control for advanced management without mandatory access control.

1.4.2.1 Hardware Specifics

All security functions claimed in this ST apply to all architectures and systems allowed via this ST. The following exceptions apply and are also marked throughout this document:

- Linux Containers are not available on POWER architecture
- Intel SMEP protection mechanism is only available on Intel x86 CPUs.

1.4.3 Compliance with STIG and other standards

The evaluated configuration draws from many standards, including the US STIG standard. It is possible to achieve full compliance with STIG in the evaluated configuration. However, to prevent violation of other configuration standards, the evaluated configuration does not claim full compliance with STIG.

1.4.4 Required Hardware and Software

The following hardware / firmware allows the installation of the TOE:

The following hardware is allowed:

- HP based on x86 64bit Intel Xeon processors:
  - HP Proliant ML series G7, Gen8, Gen9 product line
  - HP Proliant DL series G7, Gen8, Gen9 product line
  - HP ProLiant BL series G7, Gen8, Gen9 product line
  - HP ProLiant SL series G7, Gen8, Gen9 product line
- HP based on AMD64 processors:
  - HP Proliant ML series G7, Gen8 product line
  - HP Proliant DL series G7, Gen8 product line
  - HP ProLiant BL series G7, Gen8 product line
  - HP ProLiant SL series G7, Gen8 product line
- Dell based on x86 64bit Intel:
  - Dell PowerEdge R920
  - Dell PowerEdge R930
  - Dell PowerEdge T430, T630, R430, R530, R630, R730, R730xd, M630, M830, FC430, FC630, FC830, C6320, and Precision R7910
- IBM System p based on Power 8 processors providing execution environments with PowerVM:
  - Big Endian with PowerVM: Tuleta BE model number - Power 835 model 8286-41A

    - Little Endian with RHEV for Power 3.6: Power 835 model 8284-22A
- IBM System z based on z/Architecture processors:
    - zEnterprise EC12 (zEC12)
    - zEnterprise BC12 (zBC12)
    - zEnterprise 196 (z196)
    - zEnterprise 114 (z114)

The following virtual environment is allowed as an execution environment for the TOE:

- KVM on x86 hardware as provided by RHEL 7 or later
- KVM on POWER LE hardware as provided by RHEV-H 3.6 or later

All hardware must be configured using a RAM with automated error correction mechanism present. For example ECC RAM would be suitable to cover that requirement.

1.4.5 Intended Method of Use

1.4.5.1 General-purpose computing environment

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

The TOE can be configured to operate in one of two modes, Base mode and MLS mode, as defined in section 1.4.2 of this document.

In MLS mode, the TOE uses mandatory access control together with discretionary and role-based access control. In MLS mode rules are defined to assign sensitivity labels to subjects and objects and to implement the information flow mandatory access control policy modeled based on the concept of Bell-LaPadula.

In MLS mode, administrative actions are delegated to specific roles. Any user in a role that is allowed to perform administrative actions is considered an administrative user. In addition the TOE supports types that can be associated with objects and domains that can be associated with processes. Roles are defined by the domains they have access to. A predefined policy file, which is part of the TOE configuration, defines the rules between domains and types. With this definition of roles and the access rights implied by the individual roles the TOE implements the role-based access control for advanced management.

The TOE is intended to operate in a networked environment with other instantiations of the TOE as well as other well-behaved peer systems operating within the same management domain. All those systems need to be configured in accordance with a defined common security policy.

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 RHEL.

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

In MLS mode, the TOE enforces a mandatory access control policy based on sensitivity labels that are attached to objects managed by the TOE. The mechanisms to attach those labels to the objects and assign initial values to those labels are implemented in the SELinux security module which extends the security mechanisms of the Linux kernel using the loadable security module feature. SELinux provides a flexible way to define security policies to be enforced for subjects and objects within the kernel. This evaluation is based on the SELinux policy with MLS support to address the requirements of the OSPP extended package for Labeled Security and the roles and privileges required to manage this policy efficiently.

Red Hat Enterprise Linux has significant security extensions compared to standard UNIX systems:

- Access Control Lists
- Domains and type enforcement (MLS mode only)
- Labels assigned to a number of kernel objects within a security context defined and managed by the SELinux security module (MLS mode only)
- Block device encryption and ensuring the confidentiality of data at rest
- (not on POWER architecture) Strong user space isolation by providing different Linux Containers. The Linux Containers provide isolation, resource accounting and limitation and Linux kernel service limitation for different classes of objects.

1.4.5.2 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 mechanisms provided by the TOE. Such installations and usage scenarios are typical for systems accessed by processes or users local to, or with otherwise protected access to, the computer system.

Note: The TOE provides the platform for installing and running arbitrary services. These additional services are not part of the TOE. The TOE is solely the operating system which provides the runtime environment for such services.

All human users, if existent, as well as all services offered by RHEL are assigned unique user identifiers within the single host system that forms the TOE. This user identifier is used together with the attributes and roles assigned to the user identifier as the basis for access control decisions. Except for virtual machine accesses, the TOE authenticates the claimed identity of the user before allowing the user to perform any further actions. Services may be spawned by the TOE without the need for user-interaction. The TOE internally maintains a set of identifiers associated with processes, which are derived from the unique user identifier upon login of the user or from the configured user identifier for a TOE-spawned service. Some of those identifiers may change during the execution of the process according to a policy implemented by the TOE.

1.4.6 Major Security Features

The primary security features of the TOE are specified as part of the section 1.5.2.2 logical boundary description.

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

1.5 TOE Description

1.5.1 Introduction

Red Hat Enterprise Linux is a general purpose, multi-user, multi-tasking Linux based operating system. It provides a platform for a variety of applications.

The RHEL evaluation covers a potentially distributed network of systems running the evaluated versions and configurations of RHEL as well as other peer systems operating within the same management domain. The hardware platforms selected for the evaluation consist of machines which are available when the evaluation has completed and to remain available for a substantial period of time afterwards.

The TOE Security Functions (TSF) consist of functions of RHEL that run in kernel mode plus some trusted processes. 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 as with other operating system evaluations they are not considered to be part of this TSF.

The hardware, the BootProm or 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, including applications allowing access of the TOE via cryptographically protected communication channels, such as SSH.

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 SELinux security module can be configured to enforce the mandatory access control policy. The following access control rules are enforced by SELinux:

- Bell-LaPadula access control model is implemented based on labels assigned to subjects and objects (MLS mode).
- Type enforcement: As part of the SELinux label given to every object and subject in the system, a role identifier as well as type identifier can be mapped. The type identifier mapped to subjects is also called a domain. Contrary, a type identifier mapped to an object is called a type. SELinux access rules between domains and types are provided with the different SELinux access control policies. Roles are now defined as a collection of domains. Users can be assigned to one or more roles and inherit the domains assigned to the roles. Based on these domains, the subject therefore is bound to the SELinux access control rules. With this definition of roles and the access rights implied by the individual roles the TOE implements the role-based access control for advanced management (MLS mode).

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. The additional documentation specific for the evaluated configuration provides guidance how to set up such applications on the TOE in a secure way.

1.5.2 TOE boundaries

1.5.2.1 Physical

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

- Red Hat Enterprise Linux in the above mentioned version

The TOE is supplied on ISO images distributed via the Red Hat Network. The TOE includes a package holding the additional user and administrator documentation.

The general TOE documentation is also available online at the Red Hat Network.

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

- Evaluated Configuration Guide published by Red Hat at the end of the evaluation
- Manual pages for all applications, configuration files and system calls

The hardware applicable to the evaluated configuration is listed 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 (like the hypervisor state or the SMM)
- Full testing of the security functionality on all listed hardware systems

1.5.2.2 Logical

All security functions claimed in this ST apply to all architectures and systems allowed via this ST. The following exceptions apply and are also marked throughout this document:

- Linux Containers are not available on POWER architecture
- Intel SMEP protection mechanism is only available on Intel x86 CPUs.

The primary security features of the TOE are enumerated as follows:

Auditing

The Lightweight Audit Framework (LAF) is designed to be an audit system making Linux compliant with the requirements from Common Criteria. LAF is able to intercept all system calls as well as retrieving audit log entries from privileged user space applications. The subsystem allows configuring the events to be actually audited from the set of all events that are possible to be audited.

The TOE can be deployed as an audit server that receives audit logs from other TOE instances. These audit logs are stored locally. The TOE provides search and review facilities to authorized administrators for all audit logs.

Cryptographic support

The TOE provides cryptographically secured communication to allow remote entities to log into the TOE. For interactive usage, the SSHv2 protocol is provided. The TOE provides the server side as well as the client side applications. Using OpenSSH, password-based and public-key-based authentication are allowed.

In addition to OpenSSH, the TOE provides IPSec for a cryptographically secured communication with other remote entities. IPSec is offered together with IKEv1 and IKEv2 for the key negotiating aspect. The implementations of IKEv1 and IKEv2 allow a certificate based authentication of the remote peer.

To secure the communication between the TOE and remote trusted IPA authentication databases, the TOE allows the use of TLS.

In addition, the TOE provides confidentiality protected data storage using the device mapper target dm_crypt. Using this device mapper target, the Linux operating system offers administrators and users cryptographically protected block device storage space. With the help of a 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 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.

Packet filter

The TOE provides a stateless and stateful packet filter for regular IP-based communication. OSI Layer 3 (IP) and OSI layer 4 (TCP, UDP, ICMP) network protocols can be controlled using this packet filter. To allow virtual machines to communicate with the environment, the TOE provides a bridging functionality. Ethernet frames routed through bridges are controlled by a separate packet filter which implements a stateless packet filter for the TCP/IP protocol family.

The packet filtering functionality offered by the TOE is hooked into the TCP/IP stack of the kernel at different locations. Based on these locations, different filtering capabilities are applicable. The lower level protocols are covered by the EBTables filter mechanism which includes the filtering of Ethernet frames including the ARP layer -- EBTables is not covered in this evaluation. The higher level protocols of TCP/IP are covered with the IPTables mechanism which allows filtering of IP and TCP, UDP, ICMP packets. In addition, IPTables offers a stateful packet filter for the mentioned higher level protocols.

Identification and Authentication

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

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

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

The TOE provides a framework to authenticate with remote IPA servers. The SSSD daemon establishes the connection to the remote authentication stores and provides a local authentication cache in case the connection is severed. SSSD is integrated with the Linux authentication mechanism by using a PAM module.

Discretionary Access Control

DAC allows owners of named objects to control the access permissions to these objects. These owners can permit or deny access for 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.

In addition to the standard Unix-type permission bits for file system objects as well as IPC objects, the TOE implements POSIX access control lists. These ACLs allow the specification of the access to individual file system objects down to the granularity of a single user.

Mandatory Access Control

The TOE supports mandatory access control based on the following concepts:

- Sensitivity labels which are automatically attached to processes and objects. The access control policy enforced using these labels is derived from the Bell-LaPadula access control model.
- Role-based access control is implemented with roles that are defined via types and domains. A “type” is a security attribute given to an object or a process. The type of a process is commonly called a “domain”. Policy rules define how domains may interact with objects bearing types and with other domains. Roles can be assigned to users and define which user is associated with which domain. A user may have several roles assigned to him but will always act in one role only. To change from his current role to another role that has been assigned to the user, the TOE provides an application which requires re-authentication. The TOE has a hierarchical set of roles defined in the policy. Those are:
  - Root administrator: This is the classical superuser role which is hierarchical to all other roles,
  - System process: This is a role that should be assigned to specific system processes like daemons,
  - System administrator: This is a role for general system administration,
  - Security administrator: This is a role for the administration of security (policy and security contexts),
  - Audit administrator: This is a role for administration of the audit policy and the evaluation of audit records,
  - Staff: This is a user role for users allowed to use the newrole and su commands,
  - User: This is a general user role without being allowed to use the newrole and su commands,

Users cannot interfere with these labels. The TOE uses SELinux with an appropriate SELinux policy to enforce the mandatory access control.

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 remote management via OpenSSH. Administrative users can log in remotely in base as well as in MLS mode and perform the same management tasks as a locally operating administrator.

Runtime Protection Mechanisms

The TOE provides mechanisms to prevent or significantly increase the complexity of an exploitation of common buffer overflow and similar attacks. These mechanisms are used for the TSF and are available to untrusted code.

Runtime protection against programming errors: The TOE implements multiple countermeasures against exploitation of programming errors. Standard programming errors, such as buffer overflows are exploitable using a set of exploitation techniques. The TOE blocks or significantly increases the challenge to use these techniques with the following different approaches:

- Prevention of code execution on the process' or thread's stack. This prevents standard buffer overflow attacks which writes executable code (e.g. the shellcode) into a stack variable and causes the CPU to execute it.
- The technique called read-only relocation (RELRO) implemented by the Linux loader and linker marks the memory with the ELF segments holding the global object table (GOT) and procedure linking table (PLT) after the resolution by the linker but before the main() function of the application is called as read only. These sections store offset tables required for the dynamic linking mechanism and, if abused, allow attackers to modify the jump addresses of object accesses. Marking these memory segments read-only requires the dynamic linker to resolve all library symbols of shared libraries during load time of the application. Especially marking the PLT as read only incurs a significant performance penalty. Therefore, the TOE implements two types of RELRO: partial and full.
  - The full RELRO support implies that for an application and all depending shared libraries the PLT is set read only in addition to all ELF header sections other than the data segments. If at least one depending library is not compiled with full RELRO, the entire application cannot be claimed to have full RELRO. Note, the TOE code does not use full RELRO.
  - Partial RELRO implies that for an application and all depending libraries still all ELF sections except the data segments are marked read only. But the PLT is not marked read only in either the application or at least one depending shared library. If at least one depending library is compiled without RELRO, the entire application cannot be claimed to have partial RELRO.
- On Intel CPUs with appropriate support, the SMEP and SMAP functions are used which prevent the kernel from dereferencing user space memory (except for well-defined cases) and prevent the execution of code residing in user space memory.

Linux Container Framework Support (not on POWER architecture)

Linux Containers provide execution environments for processes. These Linux Containers isolate the processes, ensure resource accounting and limitation as well as Linux kernel service limitation.

The TOE offers the following mechanisms that together can be used to form a container:

- Resource accounting and limitation is implemented with Linux control groups. These control groups track processes and their children and allow resources to be assigned to and limited for these process groups.
- The TOE provides a namespace separation for different classes of objects maintained by the TOE. Within a namespace, any subject is only able to access objects associated with that namespace. Any object outside the namespace is inaccessible by subjects. Especially, the host software operation and its security behavior cannot be interfered with from software executing within a namespace. The Linux Namespaces are intended to separate processes and their resources from each other.
- Linux kernel service limitation is implemented by denying the use of system calls to a group of processes and their children. The seccomp filter implements the technical aspect of system call filtering.

The TOE provides the framework mechanisms that must be used by an application in an appropriate manner to form Linux Containers. Applications outside the TOE, such as Docker or the LXC tool set make use of those frameworks to form such Linux Containers. Though, Docker or other applications controlling such containers are not covered by any SFRs in this Security Target.

To comply with the naming schema from OSPP extended package for virtualization (which is used as a basis for the SFRs covering the Linux Container support, but not claimed), containers are also referred to as compartments.

1.5.2.3 Configurations

The evaluated configurations are defined as follows:

- The CC evaluated package set must be selected at install time in accordance with the description provided in the Evaluated Configuration Guide and installed accordingly.
- During installation time, the administrator selected either the MLS or the Base mode of operation.
- The TOE supports the use of IPv4 and IPv6, both are also supported in the evaluated configuration. IPv6 conforms to the following RFCs:
  - RFC 2460 specifying the basic IPv6 protocol
  - IPv6 source address selection as documented in RFC 3484
  - Linux implements several new socket options (IPV6_RECVPKTINFO, IPV6_PKTINFO, IPV6_RECVHOPOPTS, IPV6_HOPOPTS, IPV6_RECVDSTOPTS, IPV6_DSTOPTS, IPV6_RTHDRDSTOPTS, IPV6_RECVRTHDR, IPV6_RTHDR, IPV6_RECVHOPOPTS, IPV6_HOPOPTS, IPV6_{RECV,}TCLASS) and ancillary data in order to support advanced IPv6 applications including ping, traceroute, routing daemons and others. The following section introduces Internet Protocol Version 6 (IPv6). For additional information about referenced socket options and advanced IPv6 applications, see RFC 3542

    - Transition from IPv4 to IPv6: dual stack, and configured tunneling according to RFC 4213.
    - Additional RFCs covering various cryptographic aspects are outlined as part of the Security Functional Requirements.
- The default configuration for identification and authentication are the defined password-based PAM modules as well as by 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.

Deviations from the configurations and settings specified with the Evaluated Configuration Guide are not permitted.

The TOE comprises a single system (and optional peripherals) running the TOE software listed. Cluster configurations touching the state information of security functions are not permitted in the evaluated configuration. This means it is permissible to install applications which by themselves offer cluster functionality covering their state, such as JBoss EAP.

1.5.2.4 TOE Environment

Several 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 of the TOE systems implements its own security policy. The TOE does not include any synchronization function for those policies. As a result a single user may have user accounts on each of those systems with different UIDs, different roles, and other different attributes. (A synchronization method may optionally be used, but it not part of the TOE and must not use methods that conflict with the TOE requirements.)

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

1.5.2.5 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 or technical entity.
- Processes acting on behalf of a human user or technical entity providing a virtual machine environment.

Named objects:

- File system objects in the following allowed file systems:
    - XFS - standard file system for general data
    - VFAT - special purpose file system for UEFI BIOS support mounted at /boot/efi
    - Ext4 - standard file system for general data
    - iso9660 - ISO9660 file system for CD-ROM and DVD

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

  Note that the TOE supports a number of additional virtual (i.e. without backing of persistent storage) file systems which are only accessible to the TSF - they are not or cannot be mounted. All above mentioned virtual file systems implement access decisions based DAC attributes inferred from the underlying process' DAC attributes. Additional restrictions may apply for specific objects in this file system.
- Linux Namespace resources (implemented by namespaces for individual object classes) - (not on POWER architecture):
  - Mount namespace: all file system objects
  - User namespace: all user identifiers and group identifiers (note: Linux capabilities are handled in a special way as outlined in FDP_ACC.2(Namespaces).
  - PID namespace: all process identifiers
  - IPC namespace: all semaphores, SYSV message queues, POSIX message queues, shared memory segments
  - Network namespace:
    - IPv4 sockets and its properties including properties of higher-level protocols
    - IPv6 sockets and its properties including properties of higher-level protocols
    - unix domain sockets and its properties including properties of higher-level protocols
    - loopback sockets and its properties including properties of higher-level protocols
    - packet sockets and its properties including properties of higher-level protocols
    - packet filter configuration
    - XFRM configuration
  - UTS namespace: Naming of system (hostname, OS version, OS type, OS release)

Please note that CPU restrictions and memory range assignments are not listed as part of Linux namespace resources, because the host system scheduler automatically enforces CPU restrictions. Also, the host system virtual memory management functionality automatically enforces the memory range restriction.

- Linux control group resources (implemented by Linux control groups for individual object classes -- the trailing name in the following list indicates the technical control group name in the cgroup file system hierarchy) - (not on POWER architecture):
  - CPU accounting: cpuacct
  - Starting and stopping process groups at once (freezer control group): freezer
  - Devices: devices
  - CPU utilization (not to be mixed with the cpuset covering NUMA management): cpu
  - Memory range: memory
- Linux system call resources - (not on POWER architecture):
  - System call number
  - System call parameter
- Inter Process Communication (IPC) objects:
  - Semaphores
  - Shared memory
  - Message queues
  - Named pipes
  - UNIX domain socket special files
- Network sockets (irrespectively of their type - such as Internet sockets, netlink sockets)
- Block device objects
- at and cron job queues maintained for each user
- print job queues maintained for each user

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 the security attributes defined by FIA_ATD.1
- 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 CC Conformance Claim

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 Protection Profiles and PP packages:

- [OSPP]: BSI Operating System Protection Profile. Version 2.0 as of 2010; strict conformance.
- [OSPP-AM]: BSI OSPP Extended Package - Advanced Management. Version 2.0 as of 2010; strict conformance.
- [OSPP-LS]: BSI OSPP Extended Package - Labeled Security. Version 2.0 as of 2010; strict conformance.

Common Criteria [CC] version 3.1 revision 4 is the basis for this conformance claim.

3 Security Problem Definition

3.1 Threat Environment

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:

- 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

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.DATA_NOT_SEPARATED

The TOE might not adequately separate data on the basis of its sensitivity label, thereby allowing information to flow illicitly from or to users.

T.ACCESS.COMPENV (not on POWER architecture)

A threat agent might access the runtime environment of other compartments in an unauthorized manner.

T.COMM.COMP (not on POWER architecture)

A threat agent might access the data communicated between compartments or between a compartment and an external entity to read or modify the transferred data.

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.

3.2 Assumptions

3.2.1 Environment of use of the TOE

3.2.1.1 Physical

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.

3.2.1.2 Personnel

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.

Application note: This assumption applies to all administrative personnel and processes with capabilities operating on the TOE, including all such entities inside Linux Container.

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.

3.2.1.3 Procedural

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.IT.FUNC

The trusted IT systems executing the TOE are assumed to correctly implement the functionality required by the TSF to enforce the security functions.

A.KEYS

It is assumed that digital certificates, certificate revocation lists (CRLs) used for certificate validation, private and public keys, as well as passwords used for:

- SSH client authentication,
- SSH server authentication,
- TLS client authentication,
- TLS server authentication,
- IKE remote peer authentication,
- Password protecting the disk encryption schema

generated externally or by the TOE, meeting the corresponding standards and providing sufficient security strength through the use of appropriate key lengths and message digest algorithms. It is also assumed that Administrators verify the integrity and authenticity of digital certificates and key material before importing them into the TOE, and verifying that certificates are signed using strong hash algorithms.

3.2.1.4 Connectivity

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.

3.3 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.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.

Note: The protection of the key can be established by access permissions to the file holding the key (when using the OpenSSH client, the key file permissions are automatically verified and the key is rejected if the permissions are not restrictive), or by encrypting the key with a passphrase. Making the SSH private key available to any other user is akin to telling that user the password for password-based authentication.

P.APPROVE

Specific rights assigned to users and controlled by the TSF shall only be exercisable if approved by a second user.

P.CLEARANCE (MLS mode)

The system must limit information flow between protected resources and authorized users based on whether the user's sensitivity label is appropriate for the labeled information.

P.LABELED_OUTPUT (MLS mode)

The beginning and end of all paged, hardcopy output must be marked with sensitivity labels that properly represent the sensitivity label of the output.

P.RESOURCE_LABELS (MLS mode)

All resources accessible by subjects and all subjects must have associated labels identifying the sensitivity levels of data contained therein.

P.USER_CLEARANCE (MLS mode)

All users must have a clearance level identifying the maximum sensitivity levels of data they may access.

P.CP.ANCHOR

Users shall control the confidentiality protection anchor for their confidentiality-protected user data, and reset/replace/modify it if desired.

4 Security Objectives

4.1 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, must allow restringing such management actions to dedicated users, and must ensure that only such authorized users are able to access management 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.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.LS.CONFIDENTIALITY (MLS mode)

The TOE will control information flow between entities and resources based on the sensitivity labels of users and resources.

O.LS.PRINT (MLS mode)

The TOE will provide the capability to mark printed output with accurate labels based on the sensitivity label of the subject requesting the output.

O.LS.LABEL (MLS mode)

The TOE will provide the capability to label all subjects, and all objects accessible by subjects, to restrict information flow based on the sensitivity labels.

O.COMP.CONTAINER (not on POWER architecture)

The TOE container functionality uses the following mechanisms concurrently to implement an execution environment to confine processes to that execution environment and thus restricting access of resources and services not assigned to that execution environment. Operations in the provided execution environment shall have no effect on other instances of such execution environments or the host system. Each individual mechanism confines processes where the use of multiple layers ensures multiple lines of defense against attacks. When combining the individual mechanisms to establish a sandbox, the TOE ensures that the overall sandbox environment is resistant to stronger attacks than attacks against the isolated use of each mechanism for sandbox confinements.

The following separation mechanisms are used:

- Seccomp system call access control
- Namespace access control where all types of namespaces are used to establish that execution environment
- Linux control groups

Note: This objective defines a property of the system that cannot fully be described with functional SFRs.

O.COMP.RESOURCE_ACCESS (not on POWER architecture)

The TOE will control access of compartments to objects and resources under its control based on:

- security attributes of the objects,
- security attributes of the compartment that attempts to access the object, and
- the type of access attempted.

The rules that determine access may be based on the value of other TSF data. Access must be controlled down to individual compartments and objects.

O.COMP.IDENT (not on POWER architecture)

For each access request, the TOE is able to identify the compartment requesting to access resources, objects or information.

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.

O.CP.ANCHOR

The TOE shall allow each user to manage the trust anchor for the confidentiality protection of his own user data.

O.RUNTIME.PROTECTION

The TOE shall offer a runtime protection mechanism for applications to close attacks vectors based on the following: code execution in specific memory regions, modification of a function's return address on the stack and modification of certain program in-memory-segments.

4.2 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.IT.SYSTEM

The trusted IT systems executing the TOE supports the enforcement of the security policy.

The required functionality is detailed in section 6.1.

4.3 Security Objectives Rationale

4.3.1 Coverage

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

Objective	Threats / OSPs
O.AUDITING	P.ACCOUNTABILITY
O.CRYPTO.NET	T.ACCESS.TSFDATA T.ACCESS.TSFFUNC
O.DISCRETIONARY.ACCESS	T.ACCESS.TSFDATA T.ACCESS.USERDATA
O.NETWORK.FLOW	T.RESTRICT.NETTRAFFIC
O.SUBJECT.COM	T.ACCESS.TSFDATA T.ACCESS.USERDATA
O.I&A	T.IA.MASQUERADE T.IA.USER
O.MANAGE	T.ACCESS.TSFFUNC P.ACCOUNTABILITY P.USER
O.TRUSTED_CHANNEL	T.ACCESS.USERDATA T.ACCESS.COMM
O.ROLE.DELEGATE	T.ROLE.SNOOP T.ROLE.DELEGATE
O.ROLE.MGMT	T.ACCESS.TSFFUNC
O.ROLE.APPROVE	P.APPROVE
O.LS.CONFIDENTIALITY (MLS mode)	T.DATA_NOT_SEPARATED P.CLEARANCE (MLS mode) P.USER_CLEARANCE (MLS mode)

Objective	Threats / OSPs
O.LS.PRINT (MLS mode)	P.LABELED_OUTPUT (MLS mode)
O.LS.LABEL (MLS mode)	P.RESOURCE_LABELS (MLS mode) P.USER_CLEARANCE (MLS mode)
O.COMP.CONTAINER (not on POWER architecture)	T.ACCESS.COMPENV (not on POWER architecture) T.COMM.COMP (not on POWER architecture)
O.COMP.RESOURCE_ACCESS (not on POWER architecture)	T.ACCESS.COMPENV (not on POWER architecture) T.COMM.COMP (not on POWER architecture)
O.COMP.IDENT (not on POWER architecture)	T.ACCESS.COMPENV (not on POWER architecture) T.COMM.COMP (not on POWER architecture)
O.CP.USERDATA	T.ACCESS.CP.USERDATA
O.CP.ANCHOR	P.CP.ANCHOR
O.RUNTIME.PROTECTION	T.ACCESS.TSFDATA T.ACCESS.USERDATA

Table 1: Mapping of security objectives to threats and policies

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

Objective	Assumptions / Threats / OSPs
OE.ADMIN	A.MANAGE A.AUTHUSER A.TRAINEDUSER A.KEYS
OE.REMOTE	A.CONNECT T.ACCESS.COMM
OE.INFO_PROTECT	A.PHYSICAL A.MANAGE A.AUTHUSER A.TRAINEDUSER A.KEYS P.USER P.PROTECT_SSH_KEY
OE.INSTALL	A.MANAGE A.DETECT
OE.MAINTENANCE	A.DETECT
OE.PHYSICAL	A.PHYSICAL
OE.RECOVER	A.MANAGE A.DETECT

Objective	Assumptions / Threats / OSPs
OE.TRUSTED.IT.SYSTEM	A.PEER.MGT A.PEER.FUNC A.CONNECT
OE.IT.SYSTEM	A.IT.FUNC

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

4.3.2 Sufficiency

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

Threat	Rationale for security objectives
T.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 user 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. ● O.RUNTIME.PROTECTION requiring a runtime protection mechanism for applications to close attacks vectors based on the following: code execution in specific memory regions, modification of a function's return address on the stack and modification of certain program in-memory-segments.
T.ACCESS.USERDATA	The threat of accessing user data without proper authorization is removed by: ● O.TRUSTED_CHANNEL 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. ● O.RUNTIME.PROTECTION requiring a runtime protection mechanism for applications to close attacks vectors based on the following: code execution in specific memory regions, modification of a function's return address on the stack and modification of certain program in-memory-segments.
T.ACCESS.TSFFUNC	The threat of accessing TSF functions without proper authorization is removed by:

Threat	Rationale for security objectives
	• 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.

Threat	Rationale for security objectives
T.DATA_NOT_SEPARATED	The threat of not adequately separating data on the basis of its sensitivity label, thereby allowing information to flow illicitly from or to users, is mitigated by: ● O.LS.CONFIDENTIALITY requiring the TOE to control information flow between entities and resources, based on the sensitivity labels of users and resources.
T.ACCESS.COMPENV (not on POWER architecture)	The threat of utilizing or modifying the runtime environment of compartments executing on behalf of other users is mitigated by: ● O.COMP.RESOURCE_ACCESS (not on POWER architecture) requiring the TOE to control access of compartments to objects and resources under its control. ● O.COMP.IDENT (not on POWER architecture) requiring the TOE to identify the compartment requesting to access resources, objects or information for each access request. ● O.COMP.CONTAINER (not on POWER architecture) requiring the TOE to use the the different access control mechanisms effectively.
T.COMM.COMP (not on POWER architecture)	The threat of accessing the data communicated between compartments or between a compartment and an external entity is mitigated by: ● O.COMP.RESOURCE_ACCESS (not on POWER architecture) requiring the TOE to control access of compartments to objects and resources under its control. ● O.COMP.IDENT (not on POWER architecture) requiring the TOE to identify the compartment requesting to access resources, objects or information for each access request. ● O.COMP.CONTAINER (not on POWER architecture) requiring the TOE to use the the different access control mechanisms effectively.
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.

Table 3: Sufficiency of objectives countering threats

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

Assumption	Rationale for security objectives
A.PHYSICAL	The assumption on the IT environment to provide the TOE with appropriate physical security, commensurate with the value of the IT assets protected by the TOE is covered by:

Assumption	Rationale for security objectives
	• 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.

Assumption	Rationale for security objectives
	• 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.IT.FUNC	The assumption on trusted IT systems executing the TOE to correctly implement the functionality required by the TSF to enforce the security functions is covered by: • OE.IT.SYSTEM requiring that the trusted IT systems executing the TOE supports the enforcement of the security policy.
A.KEYS	The assumption on the use of strong keys for authentication in cryptographic protocols required by the TSF to enforce the security functions is covered by: • OE.ADMIN requiring that the administrator is sufficiently knowledgeable including in the realm of cryptography to configure and use the cryptographic protocols securely. • 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.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.

Table 4: Sufficiency of objectives holding assumptions

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

OSP	Rationale for security objectives
P.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.
P.CLEARANCE (MLS mode)	The policy to limit information flow between protected resources and authorized users based on whether the user's sensitivity label is appropriate for the labeled information is implemented by: ● O.LS.CONFIDENTIALITY requiring the TOE to control information flow between entities and resources based on the sensitivity labels of users and resources.
P.LABELED_OUTPUT (MLS mode)	The policy to provide the capability to mark printed output with accurate labels based on the sensitivity label of the user causing the output is implemented by: ● O.LS.PRINT providing the capability to mark printed output with accurate labels based on the sensitivity label of the user causing the output.
P.RESOURCE_LABELS (MLS mode)	The policy that resources accessible by subjects and all subjects must have associated labels identifying the sensitivity levels of data contained therein is implemented by: ● O.LS.LABEL providing the capability to label all subjects and all objects accessible by subjects, to restrict the information flow based on the sensitivity labels.

OSP	Rationale for security objectives
P.USER_CLEARANCE (MLS mode)	The policy that all users must have a clearance level identifying the maximum sensitivity levels of data they may access is implemented by: ● O.LS.CONFIDENTIALITY requiring the TOE to control information flow between entities and resources based on the sensitivity labels of users and resources. ● O.LS.LABEL ensuring that objects and subjects can be labeled such that the TOE can restrict information flow based on those labels.
P.CP.ANCHOR	The policy that users shall control the confidentiality protection anchor for their confidentiality-protected user data, and reset/replace/modify it if desired is implemented by: ● O.CP.ANCHOR allowing each user to manage the trust anchor for the confidentiality protection of his own user data.

Table 5: Sufficiency of objectives enforcing Organizational Security Policies

5 Extended Components Definition

The Security Target uses the extended components of FDP_RIP.3 as well as FIA_USB.2 defined by [OSPP]. They are not re-defined here again.

In addition, the Security Target defines the extended component of FCS_RNG, FDP_CDP, and FDP_ACF_NA.1 families for usage within this ST.

5.1 Class FCS: Cryptographic support

5.1.1 Random number generator (RNG)

Family behaviour

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

Component levelling

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

Management: FCS_RNG.1

The following actions could be considered for the management functions in FMT:

a) There are no management activities foreseen.

Audit: FCS_RNG.1

The following actions should be auditable if FAU_GEN Security audit data generation is included in the PP/ST:

a) Minimal: There are no actions defined to be auditable.

b) Basic: There are no actions defined to be auditable.

c) Detailed: There are no actions defined to be auditable.

5.1.1.1 FCS_RNG.1 - Random number generation

Hierarchical to: No other components.

Dependencies: No dependencies.

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

- **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:**

- **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*].**

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 Class FDP: User data protection

5.2.1 Confidentiality protection (FDP_CDP)

Component levelling

The FDP_CDP family contains only one component: FDP_CDP.1.

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

FDP_CDP.1 Confidentiality protection for data at rest, requires that the TSF ensures that the user data is stored within containers controlled by the TSF protected against accesses while the TSF are executing as well as when the TSF are not enforced.

Management: FDP_CDP.1

The following actions could be considered for the management functions in FMT:

a) Management of confidentiality protection trust anchor.

Audit: FDP_CDP.1

The following actions should be auditable if FAU_GEN Security audit data generation is included in the PP/ST:

a) Minimal: The identity of any user or subject using the data storage mechanism.
b) Basic: The identity of any unauthorised user or subject attempting to use the data exchange mechanisms.
c) Detailed: The identity of any unauthorised user or subject attempting to use the data exchange mechanisms.

5.2.1.1 FDP_CDP.1 - Confidentiality for data at rest

Hierarchical to:	No other components.
Dependencies:	[FDP_ACC.1 Subset access control, or FDP_IFC.2 Complete information flow control]

FDP_CDP.1.1 The TSF shall enforce the [assignment: **access control SFP(s) and/or information flow control SFP(s)**] to store user data at rest in containers controlled by the TSF in a manner protected from unauthorised disclosure.

Rationale

This family provides requirements that address the protection of the confidentiality of user data while it is at rest within containers controlled by the TSF. This family differs from FDP_UCT which covers the confidentiality to be maintained during the transmission of user data between the TOE and another IT product.

5.2.2 Access control function (no audit) (FDP_ACF_NA)

Component levelling

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

Management: FDP_ACF_NA.1

The following actions could be considered for the management functions in FMT:

a) See FDP_ACF.1

Audit: FDP_ACF_NA.1

The following actions should be auditable if FAU_GEN Security audit data generation is included in the PP/ST:

a) Minimal: There are no actions defined to be auditable.

b) Basic: There are no actions defined to be auditable.

c) Detailed: There are no actions defined to be auditable.

5.2.2.1 FDP_ACF_NA.1 - Access control function (no audit)

Hierarchical to:	No other components.
Dependencies:	FDP_ACC.1 Subset access control FMT_MSA.3 Static attribute initialisation

FDP_ACF_NA.1.1 (see FDP_ACF.1.1)

FDP_ACF_NA.1.2 (see FDP_ACF.1.2)

FDP_ACF_NA.1.3 (see FDP_ACF.1.3)

FDP_ACF_NA.1.4 (see FDP_ACF.1.4)

Rationale

This family is an exact copy of FDP_ACF.1 as defined in the Common Criteria Part 2 with the one difference that the audit requirements are dropped. This SFR shall cover mechanisms of the TOE that affect users, yet auditing of any actions is not appropriate as the data that can be audited is only applicable in the exact runtime context. Any attempt to apply auditable information later (which is the initial purpose of audit) provides a meaningless result.

6 Security Requirements

6.1 Security Requirements for the Operational Environment

Although CC Version 3.1 does not mandate the use of security requirements for the IT environment, it allows to define the security objectives for the IT environment to the level of detail useful for the understanding and evaluation of a TOE. In the case of Linux, the security functionality of the TOE defined in the following sections depends on the supporting functionality defined in this section. The authors of this Security Target decided to define this functionality using the structure of Security Functional Requirements.

There are several components in the IT environment that are used by the TOE to implement the security functional requirements. Those are:

- The instructions and security mechanisms provided by the underlying processor
- The cryptographic support functions offered by a subset of the x86 CPUs. The TOE analyzes the capabilities of the CPU at runtime and verifies whether the CPU provides these security mechanisms. If they are provided by the CPU, the TOE uses these mechanisms. Otherwise, the TOE reverts back to a software implementation. Although these features are implemented as instructions of the processor and therefore is part of the CPU, it has been decided by the authors of this Security Target to treat them separate from the other instructions to allow CPUs without these features.

The cryptographic processor instructions provided by the x86 CPUs are available for all programs. The claims made in this section are only for the use of those functions by the TSF. While this checks for the correct implementation of the basic cryptographic algorithms for those instructions, no claim can be made here for applications not part of the TSF that use those instructions. They may still use those instructions incorrectly or fail to protect cryptographic keys appropriately.

All SFRs listed in this section provides details to the objective OE.IT.SYSTEM.

6.1.1 General security requirements for the abstract machine

6.1.1.1 Subset access control (FDP_ACC.1(E))

FDP_ACC.1.1
The abstract machine shall enforce the memory access control policy on instructions as subjects and memory locations and processor registers as objects.

6.1.1.2 Security-attribute-based access control (FDP_ACF.1(E))

FDP_ACF.1.1
The abstract machine shall enforce the memory access control policy to objects based on the processor state (problem or supervisor).

FDP_ACF.1.2
The abstract machine shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed: access to memory locations and special registers is based on the processor state and the state of the memory management unit. Access to dedicated processor registers is allowed only if the processor is in supervisor state when the instruction accessing the register is executed.

FDP_ACF.1.3
The abstract machine shall explicitly authorize access of subjects to objects based on the following additional rules: some dedicated processor registers may be read but not modified when the instruction accessing the register is in problem mode.

FDP_ACF.1.4
The abstract machine shall explicitly deny access of subjects to objects based on the following rule: none.

Application note
The precise definition of the objects and the rules for the access control policy differ slightly depending on the processor type. Although the underlying hardware / firmware that enforces this policy is part of the IT environment, it is analyzed and tested to provide the support required for the enforcement of the TOE's self-protection. The criteria for the analysis of the high-level design require the analysis of the underlying hardware and firmware and the security functional requirements stated here are taken as the basis for this analysis.

6.1.1.3 Static attribute initialization (FMT_MSA.3(E))

FMT_MSA.3.1
The abstract machine shall enforce the memory access control policy to provide permissive default values for security attributes that are used to enforce the SFP.

FMT_MSA.3.2
The abstract machine shall allow the no role to specify alternative initial values to override the default values when an object or information is created.

Application note
The “default” values in this case are seen as the values the processor has after startup. They have to be “permissive”, because the initialization routine needs to set up the memory management unit and the device register. With respect to the hardware, there is no “role” model implemented, but the access control policy is purely based on a single attribute (“user” or “supervisor” state) that can not be managed or assigned to a “user”. The attribute changes under well-defined conditions (when the processor encounters an exception an interrupt, or when a call gate for a higher ring of privilege is called). The security requirement FMT_MSA.1 was therefore not applicable because the security attribute cannot be “managed”. For this reason, there is also no security requirement FMT_SMR.1 included, because there are no ”roles” that need to be managed or assigned to “users”. The dependency of FMT_MSA.3 to FMT_MSA.1 and FMT_SMR.1 is therefore unresolved.

6.1.2 Security requirements for CPACF

The CPACF instruction set is a feature of IBM System z processors that provides instructions to perform cryptographic operations. Those instructions are part of the general instruction set of the processor and available to programs executing in any privilege level. The instructions provide primitives for an AES implementation. No support for key management, key protection or key generation is provided. This has to be performed by the software using the instructions. The instructions are specified in the processor manual.

6.1.2.1 Cryptographic operation (CPACF) (FCS_COP.1(2E))

FCS_COP.1.1
The CPACF instruction set shall perform encryption and decryption in accordance with a specified cryptographic algorithm AES in ECB, CTR, XTS, CBC mode and cryptographic key sizes 128, 192 or 256 bit that meet the following: FIPS 197, November 6, 2001 (AES) supported by SP800-38A, SP800-38E.

FCS_COP.1.1
The CPACF instruction set shall perform encryption and decryption in accordance with a specified cryptographic algorithm three-key Triple-DES in ECB, CTR, XTS, CBC mode and cryptographic key sizes 168 bit that meet the following: SP800-67 supported by SP800-38A, SP800-38E.

FCS_COP.1.1
The CPACF instruction set shall perform encryption and decryption in accordance with a specified cryptographic algorithm SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 mode and no cryptographic key sizes that meet the following: FIPS 180-4.

6.2 TOE Security Functional Requirements

All of the following SFRs are derived from the OSPP supplemented with additional SFRs for add-on functionality.

The following table shows the SFRs for the TOE, and the operations performed on the components according to CC part 2: iteration (Iter.), refinement (Ref.), assignment (Ass.) and selection (Sel.).

Security functional group	Security functional requirement	Base security functional component	Source	Operations			
				Iter.	Ref.	Ass.	Sel.
General-purpose computing environment	FAU_GEN.1 Audit data generation		OSPP	No	No	Yes	No
	FAU_GEN.2 User identity association		OSPP	No	Yes	No	No
	FAU_SAR.1 Audit review		OSPP	No	No	Yes	No
	FAU_SAR.2 Restricted audit review		OSPP	No	No	No	No
	FAU_SEL.1 Selective audit		OSPP	No	Yes	Yes	No
	FAU_STG.1 Protected audit trail storage		OSPP	No	No	No	Yes
	FAU_STG.3 Action in case of possible audit data loss		OSPP	No	Yes	Yes	No
	FAU_STG.4 Prevention of audit data loss		OSPP	No	Yes	Yes	Yes
	FCS_CKM.1(SYM) Cryptographic key generation	FCS_CKM.1	CC Part 2	Yes	Yes	Yes	No
	FCS_CKM.1(RSA) Cryptographic key generation	FCS_CKM.1	CC Part 2	Yes	Yes	Yes	No

Security functional group	Security functional requirement	Base security functional component	Source	Operations			
				Iter.	Ref.	Ass.	Sel.
	FCS_CKM.1(DSA) Cryptographic key generation	FCS_CKM.1	CC Part 2	Yes	Yes	Yes	Yes
	FCS_CKM.1(ECDSA) Cryptographic key generation	FCS_CKM.1	CC Part 2	Yes	Yes	Yes	No
	FCS_CKM.2(NET-SSH) Cryptographic key distribution (SSHv2)	FCS_CKM.2	CC Part 2	Yes	No	Yes	No
	FCS_CKM.2(NET-IKE) Cryptographic key distribution (IKEv1 / IKEv2)	FCS_CKM.2	CC Part 2	Yes	Yes	Yes	No
	FCS_CKM.2(NET-TLS) Cryptographic key distribution (TLS)	FCS_CKM.2	CC Part 2	Yes	No	Yes	No
	FCS_CKM.4 Cryptographic key destruction		CC Part 2	No	No	Yes	No
	FCS_COP.1(NET) Cryptographic operation	FCS_COP.1	CC Part 2	Yes	No	Yes	No
	FCS_COP.1(CP) Cryptographic operation	FCS_COP.1	CC Part 2	Yes	No	Yes	No
	FCS_RNG.1(SSL-DFLT) Random number generation (Class DRG.2)	FCS_RNG.1	ECD	No	Yes	Yes	Yes
	FCS_RNG.1(SSL-FIPS) Random number generation (Class DRG.2)	FCS_RNG.1	ECD	No	Yes	Yes	Yes
	FCS_RNG.1(DM-INIT) Random number generation (Class DRG.2)	FCS_RNG.1	ECD	No	No	Yes	Yes
	FCS_RNG.1(DM-RUN) Random number generation (Class DRG.2)	FCS_RNG.1	ECD	No	No	Yes	Yes
	FCS_RNG.1(DM-FIPS) Random number generation (Class DRG.2)	FCS_RNG.1	ECD	No	No	Yes	Yes
	FCS_RNG.1(NSS) Random number generation (Class DRG.2)	FCS_RNG.1	ECD	No	No	Yes	Yes
	FDP_ACC.1(PSO) Subset access control	FDP_ACC.1	OSPP	Yes	No	Yes	No
	FDP_ACC.1(TSO) Subset access control	FDP_ACC.1	OSPP	Yes	No	Yes	No
	FDP_ACF.1(PSO) Security attribute based access control	FDP_ACF.1	OSPP	Yes	No	Yes	No
	FDP_ACF.1(TSO) Security attribute based access control	FDP_ACF.1	OSPP	Yes	No	Yes	No

Security functional group	Security functional requirement	Base security functional component	Source	Operations			
				Iter.	Ref.	Ass.	Sel.
	FDP_IFC.2(NI) Complete information flow control	FDP_IFC.2	OSPP	Yes	No	Yes	No
	FDP_IFF.1(NI-IPTables) Simple security attributes	FDP_IFF.1	OSPP	Yes	Yes	Yes	Yes
	FDP_ITC.2(BA) Import of user data with security attributes	FDP_ITC.2	OSPP	Yes	No	Yes	No
	FDP_RIP.2 Full residual information protection		OSPP	No	No	No	Yes
	FDP_RIP.3 Full residual information protection of resources		OSPP	No	No	No	Yes
	FIA_AFL.1 Authentication failure handling		OSPP	No	No	Yes	Yes
	FIA_ATD.1(HU) User attribute definition	FIA_ATD.1	OSPP	Yes	No	Yes	No
	FIA_ATD.1(TU) User attribute definition	FIA_ATD.1	OSPP	Yes	No	Yes	No
	FIA_SOS.1 Verification of secrets		OSPP	No	No	No	No
	FIA_UAU.1 Timing of authentication		OSPP	No	No	Yes	No
	FIA_UAU.5 Multiple authentication mechanisms		OSPP	No	No	Yes	No
	FIA_UAU.7 Protected authentication feedback		OSPP	No	No	No	No
	FIA_UID.1 Timing of identification		OSPP	No	No	Yes	No
	FIA_USB.2 Enhanced User-subject binding		OSPP	No	No	Yes	No
	FPT_FLS.1(FULL) Failure with preservation of secure state - full buffer overflow protection	FPT_FLS.1	CC Part 2	Yes	No	Yes	No
	FPT_FLS.1(PARTIAL) Failure with preservation of secure state - partial buffer overflow protection	FPT_FLS.1	CC Part 2	Yes	No	Yes	No
	FPT_FLS.1(INTEL) Failure with preservation of secure state - user space protecton from kernel	FPT_FLS.1	CC Part 2	Yes	Yes	Yes	No
	FPT_STM.1 Reliable time stamps		OSPP	No	No	No	No

Security functional group	Security functional requirement	Base security functional component	Source	Operations			
				Iter.	Ref.	Ass.	Sel.
	FPT_TDC.1(BA) Inter-TSF basic TSF data consistency	FPT_TDC.1	CC Part 2	Yes	No	Yes	No
	FTA_SSL.1 TSF-initiated session locking		OSPP	No	No	Yes	No
	FTA_SSL.2 User-initiated locking		OSPP	No	No	Yes	No
	FTP_ITC.1 Inter-TSF trusted channel		OSPP	No	No	Yes	Yes
Linux Container Functionality (not on POWER architecture)	FDP_ACC.2(Namespaces) Complete access control (Namespaces)	FDP_ACC.2	CC Part 2	Yes	No	Yes	No
	FDP_ACC.2(Cgroup) Complete access control (Linux control groups)	FDP_ACC.2	CC Part 2	Yes	No	Yes	No
	FDP_ACC.2(SECCOMP) Complete access control (System Call Filtering)	FDP_ACC.2	CC Part 2	Yes	No	Yes	No
	FDP_ACF.1(Namespaces) Security attribute based access control (Namespaces)	FDP_ACF.1	CC Part 2	Yes	No	Yes	No
	FDP_ACF.1(Cgroup) Security attribute based access control (Linux control groups)	FDP_ACF.1	CC Part 2	Yes	No	Yes	No
	FDP_ACF_NA.1(SECCOMP) Security attribute based access control (System Call Filtering)	FDP_ACF_NA.1	ECD	No	No	Yes	No
	FDP_ETC.2(LC) Export of user data with security attributes	FDP_ETC.2	CC Part 2	Yes	No	Yes	No
	FDP_ITC.2(LC) Import of user data with security attributes	FDP_ITC.2	CC Part 2	Yes	No	Yes	No
	FIA_UID.2(LC) User identification before any action	FIA_UID.2	CC Part 2	No	Yes	No	No
	FPT_TDC.1(LC) Inter-TSF basic TSF data consistency	FPT_TDC.1	CC Part 2	Yes	No	Yes	No
	FMT_MSA.1(Namespaces-CACP) Management of security attributes (Namespaces)	FMT_MSA.1	CC Part 2	Yes	No	Yes	Yes
	FMT_MSA.1(Cgroup-CACP) Management of security attributes (Cgroup)	FMT_MSA.1	CC Part 2	Yes	No	Yes	Yes

Security functional group	Security functional requirement	Base security functional component	Source	Operations			
				Iter.	Ref.	Ass.	Sel.
	FMT_MSA.1(SECCOMP) Management of security attributes	FMT_MSA.1	CC Part 2	Yes	No	Yes	Yes
	FMT_MSA.3(Namespace-CACP) Static attribute initialisation (Namespaces)	FMT_MSA.3	CC Part 2	Yes	No	Yes	Yes
	FMT_MSA.3(Cgroup-CACP) Static attribute initialisation (Cgroup)	FMT_MSA.3	CC Part 2	Yes	No	Yes	Yes
	FMT_MSA.3(SECCOMP) Static attribute initialisation	FMT_MSA.3	CC Part 2	Yes	No	Yes	Yes
	FMT_MTD.1(LC-COMP) Management of TSF data	FMT_MTD.1	CC Part 2	Yes	No	Yes	Yes
Confidentiality protection of data at rest	FDP_ACC.2(CP) Complete access control	FDP_ACC.2	CC Part 2	Yes	No	Yes	No
	FDP_ACF.1(CP) Security attribute based access control	FDP_ACF.1	CC Part 2	Yes	No	Yes	No
	FDP_CDP.1(CP) Confidentiality for data at rest	FDP_CDP.1	ECD	No	No	Yes	No
Management related functionality	FMT_MSA.1(PSO) Management of object security attributes	FMT_MSA.1	OSPP	Yes	No	Yes	Yes
	FMT_MSA.1(TSO) Management of object security attributes	FMT_MSA.1	OSPP	Yes	No	Yes	No
	FMT_MSA.1(CP) Management of security attributes	FMT_MSA.1	CC Part 2	Yes	No	Yes	Yes
	FMT_MSA.3(PSO) Static attribute initialisation	FMT_MSA.3	OSPP	Yes	No	Yes	No
	FMT_MSA.3(TSO) Static attribute initialisation	FMT_MSA.3	OSPP	Yes	No	Yes	No
	FMT_MSA.3(NI) Static attribute initialisation	FMT_MSA.3	OSPP	Yes	No	Yes	Yes
	FMT_MSA.3(CP) Static attribute initialisation	FMT_MSA.3	CC Part 2	Yes	No	Yes	Yes
	FMT_MSA.4(PSO) Security attribute value inheritance	FMT_MSA.4	OSPP	No	No	Yes	No
	FMT_MTD.1(AE) Management of TSF data	FMT_MTD.1	OSPP	Yes	No	Yes	No

Security functional group	Security functional requirement	Base security functional component	Source	Operations			
				Iter.	Ref.	Ass.	Sel.
	FMT_MTD.1(AS) Management of TSF data	FMT_MTD.1	OSPP	Yes	No	Yes	Yes
	FMT_MTD.1(AT) Management of TSF data	FMT_MTD.1	OSPP	Yes	No	Yes	Yes
	FMT_MTD.1(AF) Management of TSF data	FMT_MTD.1	OSPP	Yes	No	Yes	Yes
	FMT_MTD.1(NI) Management of TSF data	FMT_MTD.1	OSPP	Yes	No	Yes	Yes
	FMT_MTD.1(IAT) Management of TSF data	FMT_MTD.1	OSPP	Yes	No	Yes	No
	FMT_MTD.1(IAF) Management of TSF data	FMT_MTD.1	OSPP	Yes	No	Yes	No
	FMT_MTD.1(IAU) Management of TSF data	FMT_MTD.1	OSPP	Yes	Yes	Yes	No
	FMT_MTD.1(SSH) Management of TSF data	FMT_MTD.1	CC Part 2	Yes	No	Yes	Yes
	FMT_MTD.1(SSSD) Management of TSF data	FMT_MTD.1	CC Part 2	Yes	No	Yes	Yes
	FMT_MTD.1(SSL) Management of TSF data	FMT_MTD.1	CC Part 2	Yes	No	Yes	Yes
	FMT_MTD.1(AM-AP) Management of TSF data	FMT_MTD.1	CC Part 2	Yes	No	Yes	Yes
	FMT_MTD.1(AM-MR) Management of TSF data	FMT_MTD.1	CC Part 2	Yes	No	Yes	Yes
	FMT_MTD.1(AM-MD) Management of TSF data	FMT_MTD.1	CC Part 2	Yes	No	Yes	Yes
	FMT_MTD.1(AM-MA) Management of TSF data	FMT_MTD.1	CC Part 2	Yes	No	Yes	Yes
	FMT_MTD.1(CP-AN) Management of TSF data	FMT_MTD.1	CC Part 2	Yes	No	Yes	Yes
	FMT_MTD.1(CP-UD) Management of TSF data	FMT_MTD.1	CC Part 2	Yes	No	Yes	Yes
	FMT_REV.1(OBJ) Revocation	FMT_REV.1	OSPP	Yes	No	Yes	No
	FMT_REV.1(USR) Revocation	FMT_REV.1	OSPP	Yes	No	Yes	No

Security functional group	Security functional requirement	Base security functional component	Source	Operations			
				Iter.	Ref.	Ass.	Sel.
	FMT_SMF.1 Specification of management functions		OSPP	No	No	Yes	No
	FMT_SMR.2 Security management roles		CC Part 2	No	No	Yes	No
MLS mode	FDP_ETC.2(LS) Export of user data with security attributes	FDP_ETC.2	OSPP	Yes	No	Yes	No
	FDP_IFC.2(LS) Complete information flow control	FDP_IFC.2	OSPP	Yes	No	Yes	No
	FDP_IFF.2(LS) Hierarchical security attributes	FDP_IFF.2	OSPP	Yes	Yes	Yes	No
	FDP_ITC.1(LS) Import of user data without security attributes	FDP_ITC.1	OSPP	Yes	No	Yes	No
	FDP_ITC.2(LS) Import of user data with security attributes	FDP_ITC.2	OSPP	Yes	No	Yes	No
	FMT_MSA.1(LS) Management of security attributes	FMT_MSA.1	OSPP	Yes	Yes	Yes	Yes
	FMT_MSA.3(LS) Static attribute initialisation	FMT_MSA.3	OSPP	Yes	No	Yes	Yes
	FPT_TDC.1(LS) Inter-TSF basic TSF data consistency	FPT_TDC.1	OSPP	Yes	No	Yes	No

Table 6: SFRs for the TOE

6.2.1 General-purpose computing environment

6.2.1.1 Audit data generation (FAU_GEN.1)

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

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; and

g) **i. (MLS mode) Assignment of Users, Roles and Privileges to Roles;**

ii. (MLS mode) Deletion of Users, Roles and Privileges from Roles;

iii. (MLS mode) Creation and Deletion of Roles

.

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 (success or failure) of the event; and

b) For each audit event type, based on the auditable event definitions of the functional components included in the PP/ST;

i. User identity (if applicable); and

ii. **i. (MLS mode) sensitivity labels of subjects, objects, or information involved;**

ii. (MLS mode) for each invocation of a security function, the administrative role that invoked the security function;

iii. (MLS mode) for access control action on user data, the administrative role that invoked the action

.

6.2.1.2 User identity association (FAU_GEN.2)

FAU_GEN.2.1 For audit events resulting from actions of identified users, the TSF shall be able to associate each auditable event with the identity of the user *consisting of the user identifier and the identifier of the Linux user namespace the user is confined to if applicable* that caused the event.

Application Note: *The TOE maintains a "Login UID", which is inherited by every new process spawned. This allows the TOE to identify the "real" originator of an event, regardless if he has changed his real and / or effective and filesystem UID e. g. using the su or sudo commands or executing a setuid or setgid program. In addition, when using Linux user namespaces, the user namespace ID must be used in conjunction with the login UID as the login UID is only applicable in the realm of the user namespace.*

6.2.1.3 Audit review (FAU_SAR.1)

FAU_SAR.1.1 The TSF shall provide **the root user with the role auditadm_r (role applies only in MLS mode)** with the capability to read **all audit information** from the audit records.

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

Application Note: *The audit records are stored in ASCII format and can therefore be read with a normal editor or pager. In addition, the TOE provides specific tools that support the interpretation of the audit trail.*

Application Note: *The audit trail is stored in a file that is readable to the users with the above mentioned capabilities only.*

6.2.1.4 Restricted audit review (FAU_SAR.2)

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

Application Note: *The protection of the audit records is based on the Unix permission bit settings defined by FDP_ACC.1(PSO) supported by FDP_ACF.1(PSO). In MLS mode, the protection additionally depends on the roles defined in FMT_SMR.2.*

6.2.1.5 Selective audit (FAU_SEL.1)

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 *consisting of the user identifier and the identifier of the Linux user namespace the user is confined to if applicable* ;

c) Outcome (success or failure) of the audit event;

d) Named object identity;

e) **Access types on a particular object;**

f) **System call number;**

g) **Performing inter-field comparison rule where the specified comparison rule triggers the audit event;**

h) **arguments to system calls;**

i) **access type to file system objects (read, write, execute, change attributes);**

j) **MLS mode: Subject sensitivity label;**

k) **MLS mode: Object sensitivity label;**.

l) **MLS mode: User role**.

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.*

6.2.1.6 Protected audit trail storage (FAU_STG.1)

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

FAU_STG.1.2 The TSF shall be able to **prevent** unauthorised modifications to the audit records in the audit trail.

Application Note: *The protection of the audit records is based on the mechanisms explained in FAU_SAR.1.*

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

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

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.*

6.2.1.8 Prevention of audit data loss (FAU_STG.4)

FAU_STG.4.1 The TSF shall *be able to* **ignore the audited events** and **perform one of the following administrator-defined actions:**

a) Stop all processes that attempt to generate an audit record;

b) Switch to single user mode;

c) Halt the system;

d) Notify the administrator

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.*

6.2.1.9 Cryptographic key generation (FCS_CKM.1(SYM))

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) *AES* 128 bits,

b) *Triple-DES* 168 bits,

c) *AES* 256 bits,

d) **AES: 192 bits**

e) **HMAC-SHA-1: 160 bits**

f) **HMAC-SHA-256: 256 bits**

g) **HMAC-SHA-384: 384 bits**

h) **HMAC-SHA-512: 512 bits**

i) **PBKDF2 using SHA-1, SHA-256, SHA-384 or SHA-512 for disk encryption: key encryption key size equal to the size of the device encryption key to be protected**

that meet the following: **cryptographic key generation algorithm based on:**

a) the key agreement and key derivation function specified in FCS_CKM.2(NET-SSH) using random numbers derived from the random number generator defined in FCS_RNG.1(SSL-DFLT) for use in OpenSSH applications when FIPS 140-2 mode is not configured;

b) the key agreement and key derivation function specified in FCS_CKM.2(NET-SSH) using random numbers derived from the random number generator defined in FCS_RNG.1(SSL-FIPS) for use in OpenSSH applications when FIPS 140-2 mode is configured;

c) FCS_RNG.1(DM-RUN) random number generator for use during initialization of confidentiality-protected disks during normal operation of the TOE when FIPS 140-2 mode is not configured.

d) FCS_RNG.1(DM-INIT) for use during initialization of confidentiality-protected disks at initial installation time when FIPS 140-2 mode is not configured.

e) FCS_RNG.1(DM-FIPS) for use during initialization of confidentiality-protected disks when FIPS 140-2 mode is configured.

f) the key agreement and key derivation function specified in FCS_CKM.2(NET-IKE) using random numbers derived from the random number generator defined in FCS_RNG.1(NSS) for use in Libreswan IKE applications.

g) the key agreement and key derivation function specified in FCS_CKM.2(NET-TLS) using random numbers derived from the random number generator defined in FCS_RNG.1(NSS) for use by SSSD when FIPS 140-2 mode is not configured;

h) PBKDF2: SP800-132 section 5.4 option 2a.

6.2.1.10 Cryptographic key generation (FCS_CKM.1(RSA))

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~~*3186-4* appendix B.3 and specified cryptographic key sizes:

a) 2048 bits,

b) 1024 bits,

c) 3072 bits

d) 4096 bits

that meet the following: **U.S. NIST FIPS PUB 186-4.**

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 when FIPS 140-2 mode is not configured;*
- *FCS_RNG.1(SSL-FIPS) for use in OpenSSH applications when FIPS 140-2 mode is configured;*

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).

6.2.1.11 Cryptographic key generation (FCS_CKM.1(DSA))

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~~*3186-4* appendix B.1 and specified cryptographic key sizes:

a) **L=1024, N=160 bits;**

b) **L=2048, N=224 bits;**

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

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

that meet the following: **U.S. NIST FIPS PUB 186-4.**

Application Note:

The TOE supports the generation of DSA keys for the OpenSSH 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 when FIPS 140-2 mode is not configured;*
- *FCS_RNG.1(SSL-FIPS) for use in OpenSSH applications when FIPS 140-2 mode is configured;*

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).

6.2.1.12 Cryptographic key generation (FCS_CKM.1(ECDSA))

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: **U.S. NIST 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 when FIPS 140-2 mode is not configured;*
- *FCS_RNG.1(SSL-FIPS) for use in OpenSSH applications when FIPS 140-2 mode is configured;*

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).

6.2.1.13 Cryptographic key distribution (SSHv2) (FCS_CKM.2(NET-SSH))

FCS_CKM.2.1 The TSF shall distribute cryptographic keys in accordance with the following specified cryptographic key distribution method that meets the following:

- **a) Diffie-Hellman key agreement method with diffie-hellman-group1-sha1 defined for the SSH protocol by [RFC4253] supported by [RFC2409];**
- **b) Diffie-Hellman key agreement method with diffie-hellman-group14-sha1 defined for the SSH protocol by [RFC4253] supported by [RFC3526];**
- **c) Diffie-Hellman key agreement method with diffie-hellman-group-exchange-sha1 defined for the SSH protocol by [RFC4253] together with [RFC4419];**
- **d) Diffie-Hellman key agreement method with diffie-hellman-group-exchange-sha256 defined for the SSH protocol by [RFC4253] together with [RFC4419];**
- **e) EC Diffie-Hellman key agreement method with ecdh-sha2-nistp256 defined for the SSH protocol by [RFC4253] together with [RFC5656];**
- **f) EC Diffie-Hellman key agreement method with ecdh-sha2-nistp384 defined for the SSH protocol by [RFC4253] together with [RFC5656];**
- **g) EC Diffie-Hellman key agreement method with ecdh-sha2-nistp521 defined for the SSH protocol by [RFC4253] together with [RFC5656];**
- **h) Public DSS, RSA, ECDSA host key exchange defined for the SSH protocol by [RFC4253];**
- **i) Pseudo-Random-Function for deriving the IV, the session key and the HMAC key from the Diffie-Hellman shared secret using the hash type specified for the selected Diffie-Hellman group as defined for the SSH protocol by [RFC4253].**

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

6.2.1.14 Cryptographic key distribution (IKEv1 / IKEv2) (FCS_CKM.2(NET-IKE))

FCS_CKM.2.1 The TSF shall distribute cryptographic keys in accordance with the following specified cryptographic key distribution method that meets the following:

- **a) Diffie-Hellman key agreement method defined for the IKEv1 protocol by [RFC2409];**
- **b) Diffie-Hellman key agreement method defined for the IKEv2 protocol by [RFC5996];**

**c) Pseudo-Random-Function for deriving the IV, the session key and the HMAC key from the Diffie-Hellman shared secret using the hash type selected as part of the IKEv1 handshake by [RFC2409]; **

d) Pseudo-Random-Function for deriving the IV, the session key and the HMAC key from the Diffie-Hellman shared secret using the hash type selected as part of the IKEv2 handshake by [RFC5996];

using all the following Diffie-Hellman Oakley groups defined in [RFC2409], [RFC3526], [RFC5114]*:*

- *2 (1024-bit MODP Group)*
- *5 (1536-bit MODP Group)*
- *14 (2048-bit MODP Group)*
- *15 (3072-bit MODP Group)*
- *16 (4096-bit MODP Group)*
- *17 (6144-bit MODP Group)*
- *18 (8192-bit MODP Group)*
- *22 (1024-bit MODP Group with 160-bit Prime Order Subgroup)*
- *23 (2048-bit MODP Group with 224-bit Prime Order Subgroup)*
- *24 (2048-bit MODP Group with 256-bit Prime Order Subgroup)*

6.2.1.15 Cryptographic key distribution (TLS) (FCS_CKM.2(NET-TLS))

FCS_CKM.2.1 The TSF shall distribute cryptographic keys in accordance with the following specified cryptographic key distribution method that meets the following:

a) Diffie Hellman domain parameters or Elliptic Curve reference provided by the remote trusted TLS server.

b) RSA-based key wrapping encapsulating the pre-master secret.

c) Pseudo-Random-Function for deriving the IV, the session key and the HMAC key from the master secret using the hash type specified for TLS 1.1 by [RFC4346] or by the agreed hash type as specified for TLS 1.2 by [RFC5246].

6.2.1.16 Cryptographic key destruction (FCS_CKM.4)

FCS_CKM.4.1 The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method **of zerorization** that meets the following: **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.*

6.2.1.17 Cryptographic operation (FCS_COP.1(NET))

FCS_COP.1.1 The TSF shall perform encryption, decryption, integrity verification, peer authentication in accordance with the following cryptographic algorithms, cryptographic key sizes and applicable standards:

a) SSH communication channel encryption using the following ciphers as defined in [RFC4253]:

1. **Three-key TDES in CBC mode (3des-cbc);**
2. **AES in CBC mode (aes128-cbc, aes192-cbc, aes256-cbc);**
3. **AES in CTR mode (aes128-ctr, aes192-ctr, aes256-ctr);**
4. **AES in GCM mode (aes128-gcm, aes256-gcm);**
5. **HMAC with SHA-1 (hmac-sha1, hmac-sha1-etm@openssh.com);**
6. **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];**

b) SSH authentication of host as defined in [RFC4252]:

1. **RSA signature verification RSASSA-PKCS1-v1.5 using SHA-1 (ssh-rsa)**
2. **DSA with L=1024, N=160 signature verification using SHA-1 (ssh-dss)**
3. **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).**

c) SSH authentication of user as defined in [RFC4252]: same ciphers as specified for SSH authentication of host.

d) 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 128 bits, 192 bits and 256 bits defined by [RFC4301] and [RFC4303].**
   iii. **TDES in CBC mode with 168 bits defined by [RFC4307].**
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].**
   iii. **TDES in CBC mode with 168 bits defined by [RFC4307].**
4. **IKE SA authentication:**
   i. **HMAC SHA-1 truncated to 96 bits;**

5. **Peer authentication algorithm:**
    i. **RSA**

e) **TLS using by the following TLS cipher strings as defined by [RFC5246]:**
    i. **Key agreement Diffie-Hellman**
        i. **TLS_DHE_RSA_WITH_AES_128_CBC_SHA**
        ii. **TLS_DHE_DSS_WITH_AES_128_CBC_SHA**
        iii. **TLS_DHE_RSA_WITH_AES_256_CBC_SHA**
        iv. **TLS_DHE_DSS_WITH_AES_256_CBC_SHA**
        v. **TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA**
        vi. **TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA**
        vii. **TLS_DHE_RSA_WITH_AES_256_CBC_SHA256**
        viii. **TLS_DHE_DSS_WITH_AES_256_CBC_SHA256**
        ix. **TLS_DHE_RSA_WITH_AES_128_GCM_SHA256**
        x. **TLS_DHE_RSA_WITH_AES_128_CBC_SHA256**
        xi. **TLS_DHE_RSA_WITH_AES_256_GCM_SHA384**
        xii. **TLS_DHE_DSS_WITH_AES_128_CBC_SHA256**
        xiii. **TLS_DHE_DSS_WITH_AES_128_GCM_SHA256**
        xiv. **TLS_DHE_DSS_WITH_AES_256_GCM_SHA384**
    ii. **Key agreement EC Diffie-Hellman**
        i. **TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA**
        ii. **TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA**
        iii. **TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA**
        iv. **TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA**
        v. **TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA**
        vi. **TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA**
        vii. **TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256**
        viii. **TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256**
        ix. **TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256**
        x. **TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256**
        xi. **TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384**
        xii. **TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384**
        xiii. **TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384**
        xiv. **TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384**
    iii. **Key exchange RSA**
        i. **TLS_RSA_WITH_AES_128_CBC_SHA**
        ii. **TLS_RSA_WITH_AES_256_CBC_SHA**
        iii. **TLS_RSA_WITH_3DES_EDE_CBC_SHA**
        iv. **TLS_RSA_WITH_AES_128_GCM_SHA256**
        v. **TLS_RSA_WITH_AES_128_CBC_SHA256**

vi. **TLS_RSA_WITH_AES_256_GCM_SHA384**
vii. **TLS_RSA_WITH_AES_256_CBC_SHA256**

Application Note:

AES-NI (x86) support is disabled in the evaluated configuration.

Application Note:

On IBM System z, the CPU cryptographic support of CPACF is used as specified in the SFR in "a); 2., 3., 4., 5., 6.", "d); 1." and "d); 2.". Therefore, CPACF covers the SSH and the IPSEC protocols and not the IKE or TLS protocol.

6.2.1.18 Cryptographic operation (FCS_COP.1(CP))

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: *The list of cryptographic primitives allowed by the TOE may be reduced when booting the system in FIPS 140-2 compliant mode. The list of allowed cryptographic primitives is given in the Security Policy for the kernel crypto API FIPS 140-2 module.*

Application Note: *The list of cryptographic primitives is consistent with the requirements defined in BSI TR-02102 version 1.0, except that the XTS block chaining mode is allowed and SHA-1 is added. XTS is standardized later than the mentioned document and commonly used for disk encryption mechanisms. Furthermore, the concerns for SHA-1 regarding collisions are not considered applicable in the context of disk encryption.*

Application Note: *The master volume key (device encryption key) is encrypted with the same cipher selected for the data encryption. The key encryption key used to perform the encryption and decryption operation of the master volume key is obtained via PBKDF2 as defined in FCS_CKM.1(SYM). Although the PBKDF2 derives an ecryption key from the user's passphrase, the*

strength of that key relates to the strength of the passphrase. As passphrases typically have less entropy than random numbers, a brute force attack against the passphrase is possible in reasonable amount of time of several months.

Application Note:

AES-NI (x86) support is disabled in the evaluated configuration.

Application Note:

On IBM System z, the CPU cryptographic support of CPACF is used for all the ciphers specified in the SFR.

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

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

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 minentropy 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 **2******19** strings of bit length 128 are mutually different with probability **of more than 1 - 2******-10**.

b) DRG.2.5: The test suite A **and no other test suite** cannot distinguish the random numbers from output sequences of ideal RNGs.

Application Note:

The OpenSSH applications use the deterministic RNG from OpenSSL to generate random numbers. Every time the ssh client is invoked, 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. Note, the OpenSSL library provides two separate deterministic RNGs, the default used in normal mode and an SP800-90A CTR_DRBG with AES-256 core using a derivation function without prediction resistance compliant DRNG in FIPS 140-2 mode. This SFR covers the DRNG provided in default mode.

6.2.1.20 Random number generation (Class DRG.2) (FCS_RNG.1(SSL-FIPS))

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 minentropy 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 **2**2-19** strings of bit length 128 are mutually different with probability **of more than 1 - 2**-10**.

b) DRG.2.5: The test suite A **and no other test suite** cannot distinguish the random numbers from output sequences of ideal RNGs.

Application Note:

The OpenSSH applications use the deterministic RNG from OpenSSL to generate random numbers. Every time the ssh client is invoked, 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. Note, the OpenSSL library provides two separate deterministic RNGs, the default used in normal mode and an SP800-90A CTR_DRBG with AES-256 core using a derivation function without prediction resistance compliant DRNG in FIPS 140-2 mode. This SFR covers the DRNG provided in FIPS 140-2 mode.

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

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 minentropy 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 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: The test suite A **and no other test suite** cannot distinguish the random numbers from output sequences of ideal RNGs.

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

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 minentropy 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 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: The test suite A **and no other test suite** cannot distinguish the random numbers from output sequences of ideal RNGs.

6.2.1.23 Random number generation (Class DRG.2) (FCS_RNG.1(DM-FIPS))

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 minimum 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 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: The test suite A **and no other test suite** cannot distinguish the random numbers from output sequences of ideal RNGs.

Application Note:

In FIPS 140-2 mode, libcryptsetup uses the SP800-90A compliant DRBG (per default it is the HMAC DRBG with SHA-256 core) provided with libgcrypt which is seeded by /dev/random during normal operation of the TOE.

Application Note:

In FIPS 140-2 mode, libcryptsetup uses the SP800-90A compliant DRBG (per default it is the HMAC DRBG with SHA-256 core) provided with libgcrypt which is seeded by /dev/urandom during initial installation time of the TOE. As no other user and no attacker is assumed to be present during the the initial installation time, /dev/urandom is considered to provide the same entropy as /dev/random.

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

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 minentropy 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 initialized with a random seed **during startup of the sssd daemon** generates output for which **2**19** strings of bit length 128 are mutually different with probability **of greater than 1-2**-10**.

b) DRG.2.5: The test suite A **and no other test suite** cannot distinguish the random numbers from output sequences of ideal RNGs.

Application Note:

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

6.2.1.25 Subset access control (FDP_ACC.1(PSO))

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

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

b) Objects:

i. Persistent Storage Objects of the following type **: all file system objects defined with the Security Policy Model**;

ii. **no other storage objects**;

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

6.2.1.26 Subset access control (FDP_ACC.1(TSO))

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

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

b) Objects:

i. Transient Storage Objects of the following type **: all IPC objects defined with the Security Policy Model**;

ii. **no other storage objects**;

c) Operations: **read, receive, write, send.**

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

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

a) Subject security attributes: file system UID, file system GID, supplemental 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. ACL for specific UIDs (ACL_USER),

ii. ACL for specific GIDs (ACL_GROUP),

iii. Maximum ACL for the file system object (ACL_MASK),

iv. Permission bits for the owning UID (equals to ACL_USER_OBJ when using ACLs),

v. Permission bits for the owning GID (equals to ACL_GROUP_OBJ when using ACLs),

vi. Permission bits for "world" (equals to ACL_OTHER when using ACLs),

vii. The following permission bits: read, write, execute (for files), search (for directories),

viii. The following access rights applicable to the file system object: SAVETXT (directories), immutable (files),

d) Access control security attributes maintained for each partition holding 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:

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 (the order of the rules is applicable on a first-match basis):

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 by ACL_USER_OBJ (ACLs); or

b) ACLs: The subject's filesystem UID is identical with the UID specified with ACL_USER of the object and the requested type of access is within the permission bits defined in ACL_USER; or

c) The subject's filesystem GID or one of the subject's supplemental GIDs identical with the owning GID and the requested type of access is within the permission bits defined for the owning GID (permission bits), or by ACL_GROUP_OBJ when there is no ACL_MASK entry (ACLs), or by the ACL_MASK entry (ACLs); or

d) ACLs: The subject's filesystem GID or one of the subject's supplemental GIDs is identical with the GID specified with ACL_GROUP of the object and the requested type of access is within the permission bits defined in ACL_GROUP; or

e) The requested type of access is within the permission bits defined for "world" (permission bits) or by ACL_OTHER (ACLs).

Application Note: *The permission bits and the ACLs are inherently consistent as the TOE assigns the permission bits to ACLs when ACLs are used. Without any ACLs specified for an object, the TOE only uses the permission bits. If at least one ACL is present or when the ACL management tools are applied for objects even without any ACL set, the permission bits are interpreted as outlined above: the ACL entry of ACL_USER_OBJ contains the owning UID permission bits, the ACL entry of ACL_GROUP_OBJ contains the owning GID permission bits, and the ACL entry of ACL_OTHER contains the permission bits for "world". The ACL entries of ACL_USER_OBJ, ACL_GROUP_OBJ and ACL_OTHER are only a different representation of the permission bits to users, they are not separate attributes in addition to permission bits. The explicit specification of ACL_USER_OBJ, ACL_GROUP_OBJ and ACL_OTHER in the rule set above in addition to the permission bits is only intended to aid the evaluator or reader in understanding the overall ruleset.*

Application Note: *Due to the fact that the permission bits are an inherent part of the ACLs, there is no precedence issue between permission bits and ACLs.*

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

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 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:

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: *The no-execute flag as well as a missing execute bit in the permission bit set or ACL for the requesting user can only be considered a convenience mechanism to prevent accidental executions of files. A missing execute permission can be circumvented using the following approaches:*

- *a binary file can be opened for reading (if the access control mechanism allows reading) with the Linux loader ld-linux.so and implicitly executed. Even without a dedicated user space loader, a user can implement the logic of the loader in an application that is marked executable to use that logic for executing any file.*
- *a script file (i.e. any ASCII file starting with a Shebang can be invoked by executing the interpreter referenced in the Shebang furnishing the ASCII file to be executed as input file.*

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

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

a) Subject security attributes: effective UID, file system UID, effective GID, file system GID, supplemental 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:

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 (the order of the rules is applicable on a first-match basis):

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 supplemental GIDs 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:

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:

a) IPC object whose name is managed with a file: see FDP_ACF.1(PSO);

b) Any other IPC object: none.

6.2.1.29 Complete information flow control (FDP_IFC.2(NI))

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

a) Subjects:

i. unauthenticated external IT entities that send network data to a network interface of the TOE;

ii. **standard Linux processes** that send and receive information mediated by the TOE;

b) Information:

i. Network data routed through the TOE;

ii. **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.

6.2.1.30 Simple security attributes (FDP_IFF.1(NI-IPTables))

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;

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 TCP, UDP, ICMP

v. TCP header flags of SYN, ACK, FIN, RST, URG, PSH, TCP sequence numbers

vi. TCP sequence numbers;

Application Note: *The refinement is applied due to an obvious error in the OSPP.*

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:

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:

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) **Matching based on the state of a TCP connection, Statistical analysis matching**;

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

a) Discard the network data **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) **No other actions**.

FDP_IFF.1.4 The TSF shall explicitly authorise an information flow based on the following rules: **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: **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 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-iptables) defines different rule sets implemented by the TOE covering the FDP_IFF.1 SFR from the OSPP base.*

6.2.1.31 Import of user data with security attributes (FDP_ITC.2(BA))

FDP_ITC.2.1 The TSF shall enforce the Persistent Storage Access Control Policy, Transient Storage Access Control Policy,Network Information Flow Control, **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: **No additional importation control rules.**

6.2.1.32 Full residual information protection (FDP_RIP.2)

FDP_RIP.2.1 The TSF shall ensure that any previous information content of a resource is made unavailable upon the **allocation of the resource to** all objects, subjects or subject/object related TSF data before the resource is assigned or made available to another subject or user.

6.2.1.33 Full residual information protection of resources (FDP_RIP.3)

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

Application Note: *The subject is represented by the data structures inside the kernel forming a process: all data structures anchored in the task_struct. The user is represented by its attributes defined by FIA_ATD.1(HU).*

6.2.1.34 Authentication failure handling (FIA_AFL.1)

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

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

a) For all administrator accounts, "disable" the account for an authorized administrator configurable time period such that there can be no more than ten attempts per minute.

b) For all other accounts, disable the user logon account until it is re-enabled by the authorized administrator.

c) 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.

6.2.1.35 User attribute definition (FIA_ATD.1(HU))

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) **MLS mode: Sensitivity label;**

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

Application Note: *The SFR of FIA_ATD.1(LS) from the OSPP extended package for labeled security is merged into this SFR.*

6.2.1.36 User attribute definition (FIA_ATD.1(TU))

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) **Container: Namespace mapping and restrictions for each namespace type;**

d) **Container: Control Group mapping and restrictions for each control group type;**

e) **Container: seccomp filter restrictions;**

Application Note: *Bullet a) of this SFR relates to FDP_IFC.2(NI) and the supporting information flow control rules specified with the iterations of FDP_IFF.1. 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 ST.*

6.2.1.37 Verification of secrets (FIA_SOS.1)

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_passwdqc.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: *The Evaluated Configuration Guide contains configuration suggestions for the password quality mechanism that covers the above mentioned probability. These configuration suggestions assume the worst-case scenario when attacking these settings.*

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.*

6.2.1.38 Timing of authentication (FIA_UAU.1)

FIA_UAU.1.1 The TSF shall allow

a) the information flow covered by the Network Information Flow Control Policy;

b) **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.

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.

6.2.1.39 Multiple authentication mechanisms (FIA_UAU.5)

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) **Authentication based on remote authentication provider**

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) **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.**

e) **For username and password based authentication, the order whether the remote authentication provider or the local database is accessed is configurable. If the authentication at either the locally store credentials or at the remote authentication provider succeeds, the authenticating user is granted access.**

6.2.1.40 Protected authentication feedback (FIA_UAU.7)

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

6.2.1.41 Timing of identification (FIA_UID.1)

FIA_UID.1.1 The TSF shall allow

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.

6.2.1.42 Enhanced User-subject binding (FIA_USB.2)

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) **MLS mode: User sensitivity level that is used to enforce the Multilevel Confidentiality Information Flow Control Policy;**

h) **The user security attributes that are used to enforce the Namespace Access Control Policy**.

i) **The user security attributes that are used to enforce the Linux Control Group Access Control Policy**.

j) **The user security attributes that are used to enforce the System Call Access Control Policy**.

Application Note: *FIA_USB.1(LS) from the OSPP extended package for labeled security is merged into this SFR.*

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:

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 supplemental GIDs shall be those specified via the supplemental group membership assignment for the user entry;

d) MLS mode: The sensitivity label associated with a subject shall be within the clearance range of the user.

e) Upon instantiating a Linux Container, the namespaces, Linux control groups and seccomp filter selected by the Linux Contained management daemon is associated with the processes representing the Linux Container.

Application Note: *The various subject UIDs are all derived from the same numeric UID per user entry stored in the /etc/passwd file.*

Application Note: *The various subject GIDs except the supplemental GIDs are all derived from the same numeric GID per user entry stored in the /etc/passwd file.*

Application Note: *The subject's supplemental 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 and the group name to GID mapping is provided in /etc/group.*

Application Note: *The initial sensitivity label for each user is maintained in /etc/selinux/mls/seusers. The clearance range for users is specified in the files /etc/selinux/mls/users/*.*

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:

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_SETUID capability.

e) The set of supplemental 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.**

g) MLS mode: The sensitivity label of any subject can be changed to a label within the clearance assigned to the effective UID of that subject. This transition is restricted to subjects possessing the mlsprocwrite or mlsprocwritetoclr MLS override attributes.

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 mechanism to change the sensitivity label of subjects is implemented by writing the new label to one of the following files: /proc/<PID>/attr/{current|execve|*create} which allow the specification of the sensitivity label for the running process (current), for the process when the execve system call is triggered (execve) or the sensitivity label that is used when create the next object (*create). The same proc files also exist on a per-thread level.*

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.*

Application Note: *The Linux Container restrictions cannot be modified.*

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: **No rules**.

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

FPT_FLS.1.1 The TSF shall preserve a secure state when the following types of failures occur:

a) Stack Canary: Modification of a function return address on the process' or thread's stack to jump to previously known processor instructions by misusing the following C programming language constructs (also known as Stack Protector Strong):

1. **Functions with stack buffers larger than 8 bytes;**
2. **Functions using alloca();**
3. **Functions with local array definitions;**
4. **Functions having references to local frame addresses;**

b) RELRO: Modification of process section other than the sections that hold compile time initialized data and sections holding the mapping of all uninitialized variables

for the runtime instances of the following binaries:

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

1. **presence of the ELF program header entry of PT_GNU_STACK and the absence of the PF_X bit in the p_flags ELF header flags;**
2. **on x86 systems, presence of the ELF program header entry of PT_GNU_RELRO with the memory range information covering the following ELF sections: same as listed in FPT_FLS.1(PARTIAL) including .got.plt;**

3. **on PPC64 systems, presence of the ELF program header entry of PT_GNU_RELRO with the memory range information covering the following ELF sections: same as listed in FPT_FLS.1(PARTIAL);**
4. **on PPC64le systems, presence of the ELF program header entry of PT_GNU_RELRO with the memory range information covering the following ELF sections: same as listed in FPT_FLS.1(PARTIAL);**
5. **on S390 systems, presence of the ELF program header entry of PT_GNU_RELRO with the memory range information covering the following ELF sections: same as listed in FPT_FLS.1(PARTIAL) including .got;**

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

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

Application Note: *The stack protection is enabled when using the GCC compiler option of -fstack-protector-strong.*

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

- *The following linker option must **not** be used: "-z execstack" (gcc: "-Wl,-z,execstack").*
- *The following assembler option must **not** be used: "--execstack" (gcc: "-Wa,--execstack").*
- *Modifications of an ELF program header entry in an already compiled binary which change the PT_GNU_STACK and PF_X flags (like using the execstack(8) application) must **not** be performed.*
- *The application or library code must not contain trampolines such as nested functions pushed onto the stack which passed as pointers to functions as this would also enable the stack execution support.*

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

Application Note: *The TOE code does not use full RELRO.*

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

FPT_FLS.1.1 The TSF shall preserve a secure state when the following types of failures occur:

a) Stack Canary: Modification of a function return address on the process' or thread's stack to jump to previously known processor instructions by misusing the following C programming language constructs (also known as Stack Protector Strong):

1. **Functions with stack buffers larger than 8 bytes;**
2. **Functions using alloca();**
3. **Functions with local array definitions;**
4. **Functions having references to local frame addresses;**

b) RELRO: Modification of process sections other than the sections that hold compile time initialized data, the sections holding the mapping of all uninitialized variables, and the dynamic procedure linking table (.got.plt)

for:

i. all processes provided with the TOE executing as daemons and their depending libraries

ii. all processes with SUID flag set provided by the TOE and their depending libraries

iii. all processes with at least one file system capability set provided by the TOE and their depending libraries

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

1. **presence of the ELF program header entry of PT_GNU_STACK and the absense of the PT_X bit in the p_flags ELF header flags;**
2. **on x86 systems, presence of the ELF program header entry of PT_GNU_RELRO with the memory range information covering the following ELF sections: .tdata, .init_array, .fini_array, .ctors, .dtors, .jcr, .data.rel.ro, .dynamic, .got;**
3. **on PPC64 systems, presence of the ELF program header entry of PT_GNU_RELRO with the memory range information covering the following ELF sections: ctors, .dtors, .jcr, .data.rel.ro, .dynamic, .got**
4. **on PPC64le systems, presence of the ELF program header entry of PT_GNU_RELRO with the memory range information covering the following ELF sections: ctors, .dtors, .jcr, .data.rel.ro, .dynamic, .got**
5. **on S390 systems, presence of the ELF program header entry of PT_GNU_RELRO with the memory range information covering the following ELF sections: .interp, .note.ABI-tag, .note.gnu.build-id, .gnu.hash, .dynsym, .dynstr, .gnu.version, .gnu.version_r, .rela.dyn, .rela.plt, .init, .plt, .text, .fini, .rodata, .eh_frame_hdr, .eh_frame, .init_array, .fini_array, .jcr .dynamic**

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

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

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

6.2.1.45 Failure with preservation of secure state - user space protecton from kernel (FPT_FLS.1(INTEL))

FPT_FLS.1.1 *On an Intel x86 with SMEP support the* ~~The~~ TSF shall preserve a secure state when the following types of failures occur:

a) SMEP: Execution of code residing in user space memory by the Linux kernel;

Application Note: *This SFR applies to all all Intel-based systems listed in section 1.4.4.*

6.2.1.46 Reliable time stamps (FPT_STM.1)

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

6.2.1.47 Inter-TSF basic TSF data consistency (FPT_TDC.1(BA))

FPT_TDC.1.1 The TSF shall provide the capability to consistently interpret **the following 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 **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.

6.2.1.48 TSF-initiated session locking (FTA_SSL.1)

FTA_SSL.1.1 The TSF shall lock an interactive session to a human user maintained by the TSF after **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 access/TSF controlled display devices other than unlocking the session.

Application Note: *The management aspect of configuring the time interval is covered by FMT_MTD.1(SSL).*

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 **password based authentication**;

b) **No other events** .

6.2.1.49 User-initiated locking (FTA_SSL.2)

FTA_SSL.2.1 The TSF shall lock an 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 **password based authentication**;

b) **No other events** .

6.2.1.50 Inter-TSF trusted channel (FTP_ITC.1)

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 and disclosure using the following mechanisms:

a) Cryptographically-protected communication channel using **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-SSH) for key exchange;**
4. **Asymmetric ciphers defined in FCS_COP.1(NET) for public key encryption;**

b) Cryptographically-protected communication channel using **TLS as defined in [RFC5246] using X.509 certificates and supporting the following cipher suites defined there: Algorithms defined in FCS_COP.1(NET)**

c) Cryptographically-protected communication channel using **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 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-IKE) for use in IKE key establishment;**
5. **Asymmetric ciphers defined in FCS_COP.1(NET) for Peer Authentication;**

.

FTP_ITC.1.2 The TSF shall permit **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 **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.*

6.2.2 Linux Container Functionality (not on POWER architecture)

All SFRs in this section do not apply to the POWER architecture.

6.2.2.1 Complete access control (Namespaces) (FDP_ACC.2(Namespaces))

FDP_ACC.2.1 The TSF shall enforce the **Namespace Access Control Policy** on

a) Subjects: processes;

b) Objects: resources defined in Security Policy Model that are mapped to Linux namespaces

and all operations among subjects and objects covered by the SFP.

FDP_ACC.2.2 The TSF shall ensure that all operations between any subject controlled by the TSF and any object controlled by the TSF are covered by an access control SFP.

6.2.2.2 Complete access control (Linux control groups) (FDP_ACC.2(Cgroup))

FDP_ACC.2.1 The TSF shall enforce the **Linux Control Group Access Control Policy** on

a) Subjects: processes;

b) Objects: resources defined in Security Policy Model that are mapped to Linux control groups

and all operations among subjects and objects covered by the SFP.

FDP_ACC.2.2 The TSF shall ensure that all operations between any subject controlled by the TSF and any object controlled by the TSF are covered by an access control SFP.

6.2.2.3 Complete access control (System Call Filtering) (FDP_ACC.2(SECCOMP))

FDP_ACC.2.1 The TSF shall enforce the **System Call Access Control Policy** on

a) Subjects: processes;

b) Objects: resources defined in Security Policy Model that are mapped to system call resources

and all operations among subjects and objects covered by the SFP.

FDP_ACC.2.2 The TSF shall ensure that all operations between any subject controlled by the TSF and any object controlled by the TSF are covered by an access control SFP.

6.2.2.4 Security attribute based access control (Namespaces) (FDP_ACF.1(Namespaces))

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

- **a) Subject security attributes:**
  - **i. Memberships to all types of namespaces defined in Security Policy Model**
  - **ii. Zero or more Linux capabilities**
- **b) Object security attributes:**
  - **i. Mount namespace: list of mount points;**
  - **ii. PID namespace: process identifiers associated with PID namespace;**
  - **iii. IPC namespace: IPC objects associated with IPC namespace;**
  - **iv. Network namespace: network objects associated with network namespace;**
  - **v. UTS namespace: UTS naming data;**

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

- **a) Object accessibility for objects managed with namespaces:**
  1. **Mount namespace: a subject can only interact with the mount points and all file system objects in this mount point if that mount point is part of the mount namespace the subject requesting the interaction is associated with;**
  2. **PID namespace: a subject can only interact with another process by referring to its PID if that process is a member of the PID namespace or one of its children the subject requesting the interaction is associated with;**
  3. **IPC namespace: a subject can only interact with an IPC object if that IPC object is part of the IPC namespace the subject requesting the interaction is associated with;**
  4. **Network namespace: a subject can only interact with the network objects if that network object is part of the network namespace the subject requesting the interaction is associated with;**
  5. **UTS namespace: a subject can only interact with the system naming information object of that object is part of the UTS namespace the subject requesting the interaction is associated with;**

Application Note: *Every subject and object is always associated with a namespace. Per default, every subject and object is part of the root namespace of the respective namespace trees. Subjects with capabilities part of the root user namespace are trusted subjects.*

Application Note: *The mount namespace functionality supports the functionality of shared and slave mounts. These mount types to not fully enforce the separation capability defined in this ST and are therefore not considered by this ST. These mount types can be used and do not interfere with the mount namespace functionality that provides object isolation for different subjects.*

FDP_ACF.1.3 The TSF shall explicitly authorise access of subjects to objects based on the following additional rules: **the creation of objects within a namespace is allowed even when another object with the same properties exist in other namespaces regardless of the placement in the tree of namespaces**.

Application Note: *For example, when creating a user namespace, the same user IDs can be created and used in different namespaces. Their scope, however, is governed by the rules in FDP_ACF.1.2 to be local to the respective namespace. This applies to all other namespaces, including networking: a process can open the same port on the same interface as another process in a different network namespace -- however, in this case the interface visible in software must be connected with a different physical device (e.g. eth0 in network namespace A is a different physical device than eth0 in network namespace B -- the same applies to the loopback device).*

FDP_ACF.1.4 The TSF shall explicitly deny access of subjects to objects based on the following additional rules: **no rules**.

6.2.2.5 Security attribute based access control (Linux control groups) (FDP_ACF.1(Cgroup))

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

- **a) Subject security attributes: Assignment of a process to zero or more control group controller**
- **b) Object security attributes: For each control group controller: Resource limit configuration**

FDP_ACF.1.2 The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed: **Processes assigned to a control group controller can only access resources that are defined by the control group controller.**

FDP_ACF.1.3 The TSF shall explicitly authorise access of subjects to objects based on the following additional rules: **processes not assigned to a control group controller governing a particular resource class are not limited by the Linux Control Group Access Control Policy**.

FDP_ACF.1.4 The TSF shall explicitly deny access of subjects to objects based on the following additional rules: **no rules**.

6.2.2.6 Security attribute based access control (System Call Filtering) (FDP_ACF_NA.1(SECCOMP))

FDP_ACF_NA.1.1 The TSF shall enforce the **System Call Access Control Policy** to objects based on the following:

- **a) Subject security attributes: Definition of an access control list associated with a process which specifies:**
  1. **System call number matching rule**

2. **System call argument matching rule (optional)**
3. **Operation that is performed when matching rule is found**

b) Object security attributes:

i. **System call number**

ii. **System call arguments**

FDP_ACF_NA.1.2 The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed: **If a rule set is applicable to a process and the process invokes a system call, the rule set is evaluated for all rules that matches the invoked system call, and (optionally) matches the system call parameter. If multiple rules apply to the request, the return value for the evaluation of a given system call will always use the highest precedent value. The following precedence applies with the highest precedence value first:**

1. **SECCOMP_RET_KILL: Results in the task exiting immediately without executing the system call.**
2. **SECCOMP_RET_TRAP: Results in the kernel sending a SIGSYS signal to the triggering task without executing the system call. The program counter will be as though the system call happened. The return value register will contain a value indicating an error.**
3. **SECCOMP_RET_ERRNO: Results in the lower 16-bits of the return value being passed to userland as the errno without executing the system call.**
4. **SECCOMP_RET_TRACE: When returned, this value will cause the kernel to attempt to notify a ptrace()-based tracer prior to executing the system call. If there is no tracer present, -ENOSYS is returned to userland and the system call is not executed.**
5. **SECCOMP_RET_ALLOW: Results in the system call being executed.**

FDP_ACF_NA.1.3 The TSF shall explicitly authorise access of subjects to objects based on the following additional rules: **no rule**.

FDP_ACF_NA.1.4 The TSF shall explicitly deny access of subjects to objects based on the following additional rules: **no rules**.

6.2.2.7 Export of user data with security attributes (FDP_ETC.2(LC))

FDP_ETC.2.1 The TSF shall enforce the **Namespace Access Control Policy** when exporting user data, controlled under the SFP(s), outside of the TOE.

FDP_ETC.2.2 The TSF shall export the user data with the user data's associated security attributes.

FDP_ETC.2.3 The TSF shall ensure that the security attributes, when exported outside the TOE, are unambiguously associated with the exported user data.

FDP_ETC.2.4 The TSF shall enforce the following rules when user data is exported from the TOE: **subjects can only send data via the network interfaces associated with the network namespace assigned to the subject**.

6.2.2.8 Import of user data with security attributes (FDP_ITC.2(LC))

FDP_ITC.2.1 The TSF shall enforce the **Namespace Access Control Policy** 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: **subjects can only receive data via the network interfaces associated with the network namespace the subject is associated with**.

6.2.2.9 User identification before any action (FIA_UID.2(LC))

FIA_UID.2.1 The TSF shall require each ~~user~~*namespace a subject acting on behalf of a user is associated with* to be successfully identified before allowing any other TSF-mediated actions on behalf of that user.

Application Note: *This SFR applies to the mappings of the different types of namespaces to a process.*

6.2.2.10 Inter-TSF basic TSF data consistency (FPT_TDC.1(LC))

FPT_TDC.1.1 The TSF shall provide the capability to consistently interpret **access control related security attributes** when shared between the TSF and another trusted IT product.

FPT_TDC.1.2 The TSF shall use **the information of the physical network interface used to receive data** when interpreting the TSF data from another trusted IT product.

6.2.2.11 Management of security attributes (Namespaces) (FMT_MSA.1(Namespaces-CACP))

FMT_MSA.1.1 The TSF shall enforce the **Namespace Access Control Policy** to restrict the ability to **change_default, query, modify, delete** the security attributes **of subjects and objects covered by the SFP** to

a) User namespace: the owner of the namespace;

b) Other namespaces: the user with the capability of CAP_SYS_ADMIN outside of any user namespace.

6.2.2.12 Management of security attributes (Cgroup) (FMT_MSA.1(Cgroup-CACP))

FMT_MSA.1.1 The TSF shall enforce the **Linux Control Group Access Control Policy** to restrict the ability to **change_default, query, modify, delete, add** the security attributes **of subjects and objects covered by the SFP** to **the user with the capability of CAP_SYS_ADMIN.**

6.2.2.13 Management of security attributes (FMT_MSA.1(SECCOMP))

FMT_MSA.1.1 The TSF shall enforce the **System Call Access Control Policy** to restrict the ability to **add** the security attributes **of subjects and objects covered by the SFP** to

a) the calling process: for adding new rules applicable to this process only,

b) the kernel: when creating a new process, all security attributes are inherited from the parent process.

6.2.2.14 Static attribute initialisation (Namespaces) (FMT_MSA.3(Namespace-CACP))

FMT_MSA.3.1 The TSF shall enforce the **Namespace 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 **nobody** to specify alternative initial values to override the default values when an object or information is created.

6.2.2.15 Static attribute initialisation (Cgroup) (FMT_MSA.3(Cgroup-CACP))

FMT_MSA.3.1 The TSF shall enforce the **Linux Control Group 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 **nobody** to specify alternative initial values to override the default values when an object or information is created.

6.2.2.16 Static attribute initialisation (FMT_MSA.3(SECCOMP))

FMT_MSA.3.1 The TSF shall enforce the **System Call Access Control Policy** to provide **permissive** default values for security attributes that are used to enforce the SFP.

FMT_MSA.3.2 The TSF shall allow the **nobody** to specify alternative initial values to override the default values when an object or information is created.

6.2.2.17 Management of TSF data (FMT_MTD.1(LC-COMP))

FMT_MTD.1.1 The TSF shall restrict the ability to **initialize** the **compartment security attributes** to **any user for creating a user namespace; the subject with the Linux capability of CAP_SYS_ADMIN applicable to the namespace the administrative action applies to for creating any other namespace**.

Application Note: *This SFR applies to FIA_UID.2(LC).*

6.2.3 Confidentiality protection of data at rest

6.2.3.1 Complete access control (FDP_ACC.2(CP))

FDP_ACC.2.1 The TSF shall enforce the **Confidentiality Access Control Policy for dm-crypt** on

a) Subjects: all subjects defined with the Security Policy Model

b) Objects:

i. Persistent Storage Objects of the following type : all file system objects defined with the Security Policy Model.

and all operations among subjects and objects covered by the SFP.

FDP_ACC.2.2 The TSF shall ensure that all operations between any subject controlled by the TSF and any object controlled by the TSF are covered by an access control SFP.

6.2.3.2 Security attribute based access control (FDP_ACF.1(CP))

FDP_ACF.1.1 The TSF shall enforce the **Confidentiality Access Control Policy for dm-crypt** to objects based on the following:

a) Subject security attributes: none as all subjects maintained by the TOE are covered;

b) Persistent storage object security attributes: all persistent storage objects located on the protected block device;

c) Block device object security attributes: master volume key used to encrypt and decrypt and data processed on that block device;

d) User security attributes: passphrase that protects the master volume key using the LUKS protection mechanism.

Application Note: *The SFR mentions two different object attributes that are relevant to the security policy. The first is the master volume key used to encrypt data stored on the block device. However, file system objects (which contain the information the user wants to protect) are only covered by the encryption, if they are stored on the encrypted block device. Therefore, the storage location of the file system objects is another object security attribute as it decides about the protection status of the object.*

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

a) Access granting when TSF are active: Every user with access to the mount point of the encrypted block device is granted access when the encrypted block device is unlocked and mounted;

b) Access granting when TSF are active: Every user not in the possession of the passphrase to unlock the encrypted block device is denied access to data stored on that block device;

c) Access granting when TSF are inactive: Every user not in the possession of the passphrase to unlock the encrypted block device is denied access to data stored on that block device.

Application Note: *The TOE provides the dm_crypt mechanism as a block device encryption. When the session key for the encryption and decryption operation is provided to the kernel, the encrypted block device is unlocked. At this point, the contents - the file system - is accessible to the kernel and can be mounted. If the session key is locked, all data is encrypted on the block device with a symmetric of either Triple-DES or AES.*

FDP_ACF.1.3 The TSF shall explicitly authorise access of subjects to objects based on the following additional rules: **no explicit access authorization to any subject.**

Application Note: *When the block device is unlocked and mounted, it behaves exactly the same way as any other mounted file system. Note that any file system specific access control mechanisms like permission bits, ACLs, and SELinux-based access control are added to the protection mechanism*

FDP_ACF.1.4 The TSF shall explicitly deny access of subjects to objects based on the following additional rules: **none.**

6.2.3.3 Confidentiality for data at rest (FDP_CDP.1(CP))

FDP_CDP.1.1 The TSF shall enforce the **Confidentiality Access Control Policy for dm-crypt** to store user data at rest in containers controlled by the TSF in a manner protected from unauthorised disclosure.

6.2.4 Management related functionality

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

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

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

FMT_MSA.1.1 The TSF shall enforce the Transient Storage Object Access Control Policy to restrict the ability to modify the security attributes of the objects covered by the SFP to the owner of the object and **users with processes granted the CAP_FOWNER capability**.

6.2.4.3 Management of security attributes (FMT_MSA.1(CP))

FMT_MSA.1.1 The TSF shall enforce the **Confidentiality Access Control Policy for dm-crypt** to restrict the ability to **modify, transfer, delete** the security attributes **of the block device objects covered by the SFP** to **the owner of the object**.

Application Note: *The SFR applies to the management of the session key that encrypts the data on the block device. Only the owner, i.e. the user that is in possession of the passphrase protecting the session, is able to modify the key, to transfer it (i.e. to protect it with an additional passphrase) or to delete the session key.*

6.2.4.4 Static attribute initialisation (FMT_MSA.3(PSO))

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 the

a) root user for a global setting applied during logon;

b) each user for a setting applicable to his processes;

c) users with write permissions to a directory for setting default ACLs

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. For ACLs, the default ACL is provided for for the root directory which, in case of absence of a default ACL entry is consistent with the umask.*

6.2.4.5 Static attribute initialisation (FMT_MSA.3(TSO))

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

a) root user for a global setting applied during logon;

b) each user 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.*

6.2.4.6 Static attribute initialisation (FMT_MSA.3(NI))

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

FMT_MSA.3.2 The TSF shall allow the **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-IPTables).*

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

6.2.4.7 Static attribute initialisation (FMT_MSA.3(CP))

FMT_MSA.3.1 The TSF shall enforce the **Confidentiality Access Control Policy for dm-crypt** to provide **restrictive** default values for security attributes that are used to enforce the SFP.

FMT_MSA.3.2 The TSF shall allow **nobody** to specify alternative initial values to override the default values when an object or information is created.

Application Note: *Restrictive default values apply to the protection of the session key: the session key is created and immediately protected with a passphrase. Therefore, only the creator of the session key is initially able to access the locked block device.*

6.2.4.8 Security attribute value inheritance (FMT_MSA.4(PSO))

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

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;

d) The newly created object's ACLs are derived from the default ACL specified for the parent directory the newly created file system object is stored in, if existent. Otherwise, no ACL is set.

.

6.2.4.9 Management of TSF data (FMT_MTD.1(AE))

FMT_MTD.1.1 The TSF shall restrict the ability to query, modify the set of audited events to **processes with the capability CAP_AUDIT_CONTROL**.

Application Note: *This SFR applies to FAU_SEL.1.*

Application Note: *Using the audit tools which in turn use the netlink interface, an administrator can configure the audit rules.*

6.2.4.10 Management of TSF data (FMT_MTD.1(AS))

FMT_MTD.1.1 The TSF shall restrict the ability to clear **delete, configure the storage location** the audit storage to **the root user**.

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.*

6.2.4.11 Management of TSF data (FMT_MTD.1(AT))

FMT_MTD.1.1 The TSF shall restrict the ability to modify **add, delete** the

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

b) action when the threshold is reached

to **the root user**.

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.*

6.2.4.12 Management of TSF data (FMT_MTD.1(AF))

FMT_MTD.1.1 The TSF shall restrict the ability to modify **add, delete** the actions to be taken in case of audit storage failure to **the root user**.

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.*

6.2.4.13 Management of TSF data (FMT_MTD.1(NI))

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

a) identification of and matching of network data;

b) actions performed on the identified network data;

to **users with processes granted the CAP_NET_ADMIN capability**.

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

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

6.2.4.14 Management of TSF data (FMT_MTD.1(IAT))

FMT_MTD.1.1 The TSF shall restrict the ability to modify the threshold for unsuccessful authentication attempts to **the root user**.

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.*

6.2.4.15 Management of TSF data (FMT_MTD.1(IAF))

FMT_MTD.1.1 The TSF shall restrict the ability to re-enable the authentication to the account subject to authentication failure to **the root user**.

Application Note: *This SFR applies to FIA_AFL.1.*

Application Note: *The account locking information is stored in the directory /var/log/faillock. Using the pam_faillock 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 it.*

6.2.4.16 Management of TSF data (FMT_MTD.1(IAU))

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

a) the root user,

b) users authorized to modify their own authentication data

.

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

Application Note: *The configuration of these parameters is performed with the configuration files /etc/passwd and /etc/shadow which are writable to the root user only. The TOE also supports IPA as a remote authentication data store. In this case, the TOE does not have the ability to protect these databases. This is ensured by the assumption of A.PEER.MGT.*

6.2.4.17 Management of TSF data (FMT_MTD.1(SSH))

FMT_MTD.1.1 The TSF shall restrict the ability to **modify** the **authentication methods provided by the OpenSSH server** to **the root user**.

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.*

6.2.4.18 Management of TSF data (FMT_MTD.1(SSSD))

FMT_MTD.1.1 The TSF shall restrict the ability to **modify** the **authentication methods provided by the SSSD server** to **the root user**.

Application Note: *This SFR applies to FIA_UAU.5.*

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

6.2.4.19 Management of TSF data (FMT_MTD.1(SSL))

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

a) the root user 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: *The time interval is configured in /etc/screenrc which is writable to root only. Normal users can configure the time interval in ~/.screenrc. The screen application enforcing the session locking can be configured to execute with /etc/profile or /etc/login.csh. The root user can place screen execution commands in these Shell startup files that prevent the loading of ~/.screenrc.*

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

FMT_MTD.1.1 The TSF shall restrict the ability to **modify, delete, clear** the **management of any TSF data** to **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 of the root user 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.*

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

FMT_MTD.1.1 The TSF shall restrict the ability to **modify, delete, clear** the **assignment of roles to users down to the granularity of single users** to **the authorised identified roles** .

Application Note: *Role-based access control is implemented with the Type Enforcement configured with the SELinux policy.*

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

FMT_MTD.1.1 The TSF shall restrict the ability to **delegate, revoke delegation of** the **administrative role of root** 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.*

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

FMT_MTD.1.1 The TSF shall restrict the ability to **modify, delete, clear** the **approval of administrative actions** to **the root user** .

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

6.2.4.24 Management of TSF data (FMT_MTD.1(CP-AN))

FMT_MTD.1.1 The TSF shall restrict the ability to **modify** the **confidentiality protection anchor** to **the owner of the dm-crypt partition** .

Application Note: *The trust anchor is the passphrases that protect the master key for a dm-crypt partition.*

6.2.4.25 Management of TSF data (FMT_MTD.1(CP-UD))

FMT_MTD.1.1 The TSF shall restrict the ability to **enable, disable** the **security attributes governing the enforcement of the Confidentiality Access Control Polity on an object** to **the owner of the dm-crypt partition** .

6.2.4.26 Revocation (FMT_REV.1(OBJ))

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

a) DAC permissions: owners of the object and authorized administrator;

b) Other security attributes: authorized administrator.

Application Note: *The privileges that constitute an authorized administrator are defined in the above mentioned FMT_* SFRs which specify the privileges needed to modify object security attributes. The same privileges are required to revoke these security attributes.*

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) **no specification of other revocation rules**.

Application Note: *Revocation of security attributes for named objects imply the revocation of access granted to users other than the owner of the object. Note that the DAC ownership management (which can be also considered as a form of access revocation) is specified in FMT_MSA.1(PSO).*

Application Note: *Sensitivity labels cannot be revoked, they can only be modified as defined by FMT_MSA.1(LS). This is consistent with the requirement that all subjects and objects must always bear a label. Therefore, this SFR covers the modification of the sensitivity label which may revoke access for subjects or users to objects.*

6.2.4.27 Revocation (FMT_REV.1(USR))

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 **authorized administrators** .

Application Note: *The privileges that constitute an authorized administrator are defined in the above mentioned FMT_* SFRs which specify the privileges needed to modify object security attributes. The same privileges are required to revoke these security attributes.*

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) **No other rules**

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

6.2.4.28 Specification of management functions (FMT_SMF.1)

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 Network Information Flow Control Policy;

e) Management of identification and authentication policy;

f) Management of user security attributes;

g) **Management of Compartment Access Control Policy;**

h) **Management of Compartment Information Flow Control Policy;**

i) **Management of Linux Container configurations;**

j) **Management of Multilevel Confidentiality Information Flow Control Policy.**

Application Note: *The given list is kept generic intentionally. This ST specifies one iteration of FMT_MTD.1 per management function required by an SFR. For each FMT_MTD.1 iteration, a corresponding application note refers to the covered SFR(s).*

6.2.4.29 Security management roles (FMT_SMR.2)

FMT_SMR.2.1 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. **no other rights;**

b) **Configurations stored by user space: administrative users defined by the access permissions to the configurations mentioned in the other management SFRs;**

c) **Functions provided by the kernel: administrative users defined by capabilities mentioned in other management SFRs;**

d) **(MLS mode only) Role-based access control: set of administrative roles for the role-based access control.**

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

a) (MLS mode only) Role of object owners: Object Owners can modify security attributes for only the objects they own (except for the sensitivity label);

b) (MLS mode only) The set of RBAC administrative roles: The set of RBAC administrative roles can modify security attributes for all objects under the control of TOE (since they automatically inherit the privileges of all Object Owners);

c) (non-MLS mode only) No role-based access control

are satisfied.

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: *Subjects possessing capabilities are still restricted by the MAC policy. To perform administrative actions, the administrative user must possess the following privileges:*

- *Respective capability;*
- *Possessing one or more MLS override attributes to perform operations which are generally denied by the MAC policy;*
- *Invocation of the "newrole" command to switch the sensitivity label for obtaining write access to system configuration files which are protected by a sensitivity label that is not equal to the sensitivity label of subjects. Please note that an additional access control mechanism is enforced in addition to the MAC policy which is completely disregarded in this ST. This additional access control mechanism (called Type Enforcement that is defined with the SELinux policy which is combined with the MLS policy) adds additional restrictions on top of the MLS policy. In order to perform administrative tasks, the newrole application must also be used to switch the subject type and role covered by the Type Enforcement mechanism.*

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

6.2.5 MLS mode

6.2.5.1 Export of user data with security attributes (FDP_ETC.2(LS))

FDP_ETC.2.1 The TSF shall enforce the **Multilevel Confidentiality Information Flow Control Policy** when exporting user data, controlled under the SFP(s), outside of the TOE.

FDP_ETC.2.2 The TSF shall export the user data with the user data's associated security attributes.

FDP_ETC.2.3 The TSF shall ensure that the security attributes, when exported outside the TOE, are unambiguously associated with the exported user data.

FDP_ETC.2.4 The TSF shall enforce the following rules when user data is exported from the TOE:

a) When data is exported in hardcopy form, each page shall be marked with a printed representation of the sensitivity label of the subject requesting the export of the page. By default, this marking shall appear on both the top and bottom of each printed page.

b) When the data is exported to a device, the security attributes shall be exported with the data using either:

1. **extended tar archive format storing extended attributes with file system objects,**
2. **CIPSO network protocol, or**
3. **IPSEC network protocol with an extension to establish the label during the IKE-based key negotiation.**

6.2.5.2 Complete information flow control (FDP_IFC.2(LS))

FDP_IFC.2.1 The TSF shall enforce the **Multilevel Confidentiality Information Flow Control Policy** on

a) Subjects: all subjects defined with the Security Policy Model;

b) Objects: all named objects defined with the Security Policy Model

and all operations that cause that information to flow among them.

FDP_IFC.2.2 The TSF shall ensure that all operations that cause any information in the TOE to flow among untrusted subjects and named objects in the TOE are covered by the Multilevel Confidentiality Information Flow Control Policy.

6.2.5.3 Hierarchical security attributes (FDP_IFF.2(LS))

FDP_IFF.2.1 The TSF shall enforce the **Multilevel Confidentiality Information Flow Control Policy** based on the following types of subject and object security attributes:

a) Subject security attributes:

i. Sensitivity label of the subject consisting of at least 8 site-definable hierarchical levels and a set of 60 site definable non-hierarchical categories;

b) Object security attributes:

i. the sensitivity label of the object consisting of at least 8 site-definable hierarchical levels and a set of 60 site definable non-hierarchical categories;

FDP_IFF.2.2 The TSF shall permit an information flow between a controlled subject and controlled object via a controlled operation if the following rules, based on the ordering relationships between security attributes hold:

a) If the sensitivity label of the subject is greater than or equal to the sensitivity label of the object, then the flow of information from the object to the subject is permitted (a read operation);

b) If the sensitivity label of the object is equal to the sensitivity label of the subject; then the flow of information from the subject to the object is permitted (a write operation);

c) If the information flow is between objects, the sensitivity label of the destination object must be greater than or equal to the sensitivity label of the source object.

Application Note: *The TOE only allows the write operation if the labels are equal. As this is more restrictive that the PP specification, the refinement is considered to be in line with the protection profile specification.*

FDP_IFF.2.3 The TSF shall enforce the **no additional rules**.

FDP_IFF.2.4 The TSF shall explicitly authorise an information flow based on the following rules:

a) MLS-override attributes assigned to a subject allow that subject to perform the operation the MLS-override attribute applies to irrespectively of the sensitivity labels of the subject or object;

b) MLS-override attributes assigned to an object allow every subject to perform the operation the MLS-override attribute applies to with that object irrespectively of the sensitivity labels of the subject or object.

.

Application Note:

The following MLS override attributes are defined (note that for most of the below-mentioned attributes the TOE also defines a twin-attribute with the same override-capability which is only applicable when additional restrictions are met - the names of these attributes are identical to their corresponding attribute listed below, extended with the suffix "toclr"):

mlsfdshare
The policy disallows the sharing of file descriptors between levels unless the file descriptor is authorized to be shared among levels.

mlsfduse
The policy disallows the sharing of file descriptors between levels unless the process is authorized to shared it among levels.

mlsfiledowngrade
Make specified domain MLS trusted for lowering the level of files.

mlsfileread
Make specified domain MLS trusted for reading from files at higher levels.

mlsfileupgrade
Make specified domain MLS trusted for raising the level of files.

mlsfilewrite
Make specified domain MLS trusted for writing to files at lower levels.

mlsfilewriteinrange
This attribute has the same meaning as mlsfilewritetoclr.

mlsipcread
Make specified domain MLS trusted for reading from System V IPC objects at any level.

mlsipcwrite
Make specified domain MLS trusted for writing to System V IPC objects at any level.

mlsnetread
Make specified domain MLS trusted for reading from sockets at any level.

mlsnetrecvall
Make specified domain MLS trusted for receiving network data from network interfaces or hosts at any level.

mlsnetwrite
Make specified domain MLS trusted for writing to sockets at any level.

mlsnetwriteranged
Same as mlsnetwritetoclr with even more restrictions on the levels of the process and the target object.

mlsprocread
Make specified domain MLS trusted for reading attributes from processes at higher levels like reading capabilities or scheduling information or performing the ptrace operation.

mlsprocsetsl
Make specified domain MLS trusted for setting the level of processes it executes.

mlsprocwrite
Make specified domain MLS trusted for writing to processes at lower levels like sending signals, setting capabilities, setting the SELinux labels for a process in the proc file.

mlsrangetrans
Make specified domain a target domain for MLS range transitions that change the current level.

mlstrustedobject
Make specified object MLS trusted and exclude it from the MLS checks.

privrangetrans
Allow the specified domain to do a MLS range transition that changes the current level.

Note: The MLS policy specifies additional MLS override attributes. However, those do not cover any objects present in the TOE as they are intended for applications using the SELinux policy in addition to the kernel (such as X11 or databases) - none of these applications are installed in the TOE.

FDP_IFF.2.5 The TSF shall explicitly deny an information flow based on the following rules: **trusted subjects with MLS override capabilities can access objects without being restricted by subject and object labels. Trusted objects can be accessed by any subject**.

FDP_IFF.2.6 The TSF shall enforce the following relationships for any two valid information flow control security attributes:

a) There exists an ordering function that, given two valid security attributes, determines if the security attributes are equal, if one security attribute is greater than the other, or if the security attributes are incomparable *with the following properties:*

i. Sensitivity labels are equal if the hierarchical levels of both labels are equal and the non-hierarchical category sets are identical;

ii. Sensitivity label A is greater than sensitivity label B if the hierarchical level of A is greater than or equal to the hierarchical level of B, and the non- hierarchical category set of A is identical to or a superset of the non-hierarchical category set of B;

iii. Sensitivity labels are incomparable if they are not equal and neither label is greater than the other as defined in 1 and 2 above;

and

b) There exists a “least upper bound” in the set of security attributes, such that, given any two valid security attributes, there is a valid security attribute that is greater than or equal to the two valid security attributes; and

c) There exists a “greatest lower bound” in the set of security attributes, such that, given any two valid security attributes, there is a valid security attribute that is not greater than the two valid security attributes.

6.2.5.4 Import of user data without security attributes (FDP_ITC.1(LS))

FDP_ITC.1.1 The TSF shall enforce the **Multilevel Confidentiality Information Flow Control Policy** when importing unlabeled user data controlled under the SFP, from outside of the TOE.

FDP_ITC.1.2 The TSF shall ignore any label-related security attributes associated with the unlabeled user data when imported from outside the TOE.

FDP_ITC.1.3 The TSF shall enforce the following rules when importing unlabeled user data controlled under the SFP from outside the TOE:

a) When importing unlabeled data, the TSF shall allow the

1. **process possessing the CAP_SYS_ADMIN capability and access permissions to the mount point for performing the mount operation;**
2. **processes possessing MLS override attributes;**
3. **unprivileged processes possessing no specific MLS;**
4. **import of unlabeled network data;**

to specify that the data is to be labeled with:

1. **Mount operation: applying the label of the mount point if the mounted file system does not support labels;**
2. **Processes possessing MLS override attributes: labels according to the rules allowed by the MLS override attribute specification;**
3. **Unprivileged processes: the process' own label;**
4. **Unlabeled data: the networking stack automatically labels any unlabeled data with the label configured by the administrator.**

6.2.5.5 Import of user data with security attributes (FDP_ITC.2(LS))

FDP_ITC.2.1 The TSF shall enforce the **Multilevel Confidentiality Information Flow Control Policy** when importing labeled user data, controlled under the SFP, from outside of the TOE.

FDP_ITC.2.2 The TSF shall use the label-related security attributes associated with the imported labeled 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 label-related 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: **Devices used to import data with security attributes cannot be used to import data without security attributes unless the change in device state is performed manually and is auditable.**

6.2.5.6 Management of security attributes (FMT_MSA.1(LS))

FMT_MSA.1.1 The TSF shall enforce the **Multilevel Confidentiality Information Flow Control Policy** to restrict the ability to **modify** the *label-related object* security attributes ~~[assignment: list of security attributes]~~ to **the role allowed to modify sensitivity labels of objects** .

6.2.5.7 Static attribute initialisation (FMT_MSA.3(LS))

FMT_MSA.3.1 The TSF shall enforce the **Multilevel Confidentiality Information Flow 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 **the authorised identified roles, or users that satisfy the following rules: users in an administrative role values** to specify alternative initial values to override the default values when an object or information is created.

6.2.5.8 Inter-TSF basic TSF data consistency (FPT_TDC.1(LS))

FPT_TDC.1.1 The TSF shall provide the capability to consistently interpret **label-related security attributes** when shared between the TSF and another trusted IT product.

FPT_TDC.1.2 The TSF shall use **the following network protocols to communicate labels:**

a) CIPSO;

b) IPSEC with an extension to establish the label during the IKE-based key negotiation

when interpreting the TSF data from another trusted IT product.

6.3 Security Functional Requirements Rationale

6.3.1 Coverage

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

Security functional requirements	Objectives
FAU_GEN.1	O.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-SSH)	O.CRYPTO.NET
FCS_CKM.2(NET-IKE)	O.CRYPTO.NET
FCS_CKM.2(NET-TLS)	O.CRYPTO.NET

Security functional requirements	Objectives
FCS_CKM.4	O.CRYPTO.NET
FCS_COP.1(NET)	O.CRYPTO.NET
FCS_COP.1(CP)	O.CP.USERDATA
FCS_RNG.1(SSL-DFLT)	O.CRYPTO.NET
FCS_RNG.1(SSL-FIPS)	O.CRYPTO.NET
FCS_RNG.1(DM-INIT)	O.CP.USERDATA
FCS_RNG.1(DM-RUN)	O.CP.USERDATA
FCS_RNG.1(DM-FIPS)	O.CP.USERDATA
FCS_RNG.1(NSS)	O.CRYPTO.NET
FDP_ACC.1(PSO)	O.DISCRETIONARY.ACCESS
FDP_ACC.1(TSO)	O.SUBJECT.COM
FDP_ACF.1(PSO)	O.DISCRETIONARY.ACCESS
FDP_ACF.1(TSO)	O.SUBJECT.COM
FDP_IFC.2(NI)	O.NETWORK.FLOW
FDP_IFF.1(NI-IPTables)	O.NETWORK.FLOW
FDP_ITC.2(BA)	O.DISCRETIONARY.ACCESS, O.NETWORK.FLOW, O.SUBJECT.COM
FDP_RIP.2	O.AUDITING, O.CRYPTO.NET, O.DISCRETIONARY.ACCESS, O.I&A, O.NETWORK.FLOW, O.SUBJECT.COM
FDP_RIP.3	O.AUDITING, O.CRYPTO.NET, O.DISCRETIONARY.ACCESS, O.I&A, O.NETWORK.FLOW, O.SUBJECT.COM
FIA_AFL.1	O.I&A
FIA_ATD.1(HU)	O.I&A, O.LS.LABEL (MLS mode)
FIA_ATD.1(TU)	O.NETWORK.FLOW
FIA_SOS.1	O.I&A

Security functional requirements	Objectives
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, O.LS.LABEL (MLS mode)
FPT_FLS.1(FULL)	O.RUNTIME.PROTECTION
FPT_FLS.1(PARTIAL)	O.RUNTIME.PROTECTION
FPT_FLS.1(INTEL)	O.RUNTIME.PROTECTION
FPT_STM.1	O.AUDITING
FPT_TDC.1(BA)	O.DISCRETIONARY.ACCESS, O.NETWORK.FLOW, O.SUBJECT.COM
FTA_SSL.1	O.I&A
FTA_SSL.2	O.I&A
FTP_ITC.1	O.TRUSTED_CHANNEL
FDP_ACC.2(Namespaces)	O.COMP.CONTAINER (not on POWER architecture), O.COMP.RESOURCE_ACCESS (not on POWER architecture)
FDP_ACC.2(Cgroup)	O.COMP.CONTAINER (not on POWER architecture), O.COMP.RESOURCE_ACCESS (not on POWER architecture)
FDP_ACC.2(SECCOMP)	O.COMP.CONTAINER (not on POWER architecture), O.COMP.RESOURCE_ACCESS (not on POWER architecture)
FDP_ACF.1(Namespaces)	O.COMP.CONTAINER (not on POWER architecture), O.COMP.RESOURCE_ACCESS (not on POWER architecture)
FDP_ACF.1(Cgroup)	O.COMP.CONTAINER (not on POWER architecture), O.COMP.RESOURCE_ACCESS (not on POWER architecture)
FDP_ACF_NA.1(SECCOMP)	O.COMP.CONTAINER (not on POWER architecture), O.COMP.RESOURCE_ACCESS (not on POWER architecture)
FDP_ETC.2(LC)	O.COMP.RESOURCE_ACCESS (not on POWER architecture)

Security functional requirements	Objectives
FDP_ITC.2(LC)	O.COMP.RESOURCE_ACCESS (not on POWER architecture)
FIA_UID.2(LC)	O.COMP.IDENT (not on POWER architecture)
FPT_TDC.1(LC)	O.COMP.RESOURCE_ACCESS (not on POWER architecture)
FMT_MSA.1(Namespaces-CACP)	O.COMP.RESOURCE_ACCESS (not on POWER architecture)
FMT_MSA.1(Cgroup-CACP)	O.COMP.RESOURCE_ACCESS (not on POWER architecture)
FMT_MSA.1(SECCOMP)	O.COMP.RESOURCE_ACCESS (not on POWER architecture)
FMT_MSA.3(Namespace-CACP)	O.COMP.RESOURCE_ACCESS (not on POWER architecture)
FMT_MSA.3(Cgroup-CACP)	O.COMP.RESOURCE_ACCESS (not on POWER architecture)
FMT_MSA.3(SECCOMP)	O.COMP.RESOURCE_ACCESS (not on POWER architecture)
FMT_MTD.1(LC-COMP)	O.COMP.RESOURCE_ACCESS (not on POWER architecture)
FDP_ACC.2(CP)	O.CP.USERDATA
FDP_ACF.1(CP)	O.CP.USERDATA
FDP_CDP.1(CP)	O.CP.USERDATA
FMT_MSA.1(PSO)	O.MANAGE
FMT_MSA.1(TSO)	O.MANAGE
FMT_MSA.1(CP)	O.CP.USERDATA
FMT_MSA.3(PSO)	O.MANAGE
FMT_MSA.3(TSO)	O.MANAGE
FMT_MSA.3(NI)	O.MANAGE
FMT_MSA.3(CP)	O.CP.USERDATA
FMT_MSA.4(PSO)	O.MANAGE
FMT_MTD.1(AE)	O.MANAGE
FMT_MTD.1(AS)	O.MANAGE
FMT_MTD.1(AT)	O.MANAGE

Security functional requirements	Objectives
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(SSSD)	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_MTD.1(CP-AN)	O.CP.ANCHOR
FMT_MTD.1(CP-UD)	O.CP.USERDATA
FMT_REV.1(OBJ)	O.MANAGE
FMT_REV.1(USR)	O.MANAGE
FMT_SMF.1	O.MANAGE
FMT_SMR.2	O.MANAGE
FDP_ETC.2(LS)	O.LS.CONFIDENTIALITY (MLS mode), O.LS.PRINT (MLS mode)
FDP_IFC.2(LS)	O.LS.CONFIDENTIALITY (MLS mode)
FDP_IFF.2(LS)	O.LS.CONFIDENTIALITY (MLS mode)
FDP_ITC.1(LS)	O.LS.CONFIDENTIALITY (MLS mode), O.LS.LABEL (MLS mode)
FDP_ITC.2(LS)	O.LS.CONFIDENTIALITY (MLS mode), O.LS.LABEL (MLS mode)
FMT_MSA.1(LS)	O.LS.LABEL (MLS mode)
FMT_MSA.3(LS)	O.LS.LABEL (MLS mode)

Security functional requirements	Objectives
FPT_TDC.1(LS)	O.LS.CONFIDENTIALITY (MLS mode), O.LS.LABEL (MLS mode)

Table 7: Mapping of security functional requirements to security objectives

6.3.2 Sufficiency

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

Security objectives	Rationale
O.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)] where the functionality is based on the random number generator as defined by [FCS_RNG.1(SSL-DFLT), FCS_RNG.1(SSL-FIPS), 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-SSH), FCS_CKM.2(NET-IKE), FCS_CKM.2(NET-TLS)]. 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.ACCESS	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(BA), FPT_TDC.1(BA)]. The protection of reused resources ensures that no data leaks from other protected sources [FDP_RIP.2, FDP_RIP.3].

Security objectives	Rationale
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-IPTables)]. 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(BA), FPT_TDC.1(BA)]. 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(BA), FPT_TDC.1(BA)]. 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), FMT_MTD.1(SSSD)]. ● the session locking threshold [FMT_MTD.1(SSL)].

Security objectives	Rationale
	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.2]. 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.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.LS.CONFIDENTIALITY (MLS mode)	The information flow control policy is defined by specifying the subjects, objects, security attributes and rules in [FDP_IFC.2(LS), FDP_IFF.2(LS)]. Supportive to the enforcement of the policy are the automated label assignment when exporting data [FDP_ETC.2(LS)] and during the import of data [FDP_ITC.1(LS), FDP_ITC.2(LS)]. For assigning labels to imported data, the label information transmitted with the data must be interpretable by the TOE [FPT_TDC.1(LS)].
O.LS.PRINT (MLS mode)	The addition of label information on exported data during printing is governed by [FDP_ETC.2(LS)].
O.LS.LABEL (MLS mode)	The assignment of labels to users is performed during user-subject binding [FIA_USB.2] with security attributes maintained by the TOE [FIA_ATD.1(HU)]. Object labels are assigned to objects when importing them into the TOE [FDP_ITC.1(LS), FDP_ITC.2(LS), FPT_TDC.1(LS)]. The management of labels is allowed for the TOE with [FMT_MSA.1(LS), FMT_MSA.3(LS)].
O.COMP.CONTAINER (not on POWER architecture)	The different access control policies for the resources belonging to the different sandboxes is defined with [FDP_ACC.2], [FDP_ACF.1].
O.COMP.RESOURCE_ACCESS (not on POWER architecture)	The access control policy for the resources belonging to the different compartments is defined with [FDP_ACC.2(Namespaces), FDP_ACC.2(Cgroup)], [FDP_ACF.1(Namespaces), FDP_ACF.1(Cgroup)]. As the TOE shall allow export of data belonging to compartments, the TOE assigns the security attributes for enforcing the access control policy to the communicated data as specified with [FDP_ETC.2(LC)], [FDP_ITC.2(LC)], and [FPT_TDC.1(LC)].

Security objectives	Rationale
	The TOE allows the reduction of the attack surface the kernel offers to compartments by preventing system calls to be executed by compartments. The specification of the access control of which system call is given with [FDP_ACC.2(SECCOMP)] and [FDP_ACF_NA.1(SECCOMP)]. Management of the security attributes for the access control policy is specified with the iterations of [FMT_MSA.1], [FMT_MSA.3] as well as FMT_MTD.1(LC-COMP).
O.COMP.IDENT (not on POWER architecture)	The identification of compartments to support the information flow control and access control policies is established with [FIA_UID.2(LC)].
O.CP.USERDATA	The confidentiality protection mechanism for user data at rest is provided with the access control policy specified with [FDP_ACC.2] and [FDP_ACF.1] and supported by the cryptographic operations defined in [FCS_COP.1(CP)]. In addition, the confidentiality mechanism is defined with [FDP_CDP.1]. The 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), FCS_RNG.1(DM-FIPS)]. The management of the confidentiality protection mechanism is covered by [FMT_MSA.1] and [FMT_MSA.3] covering the general management aspects. [FMT_MTD.1(CP-UD)] allows owners of user data to select which of their data is covered by the confidentiality protection mechanism.
O.CP.ANCHOR	The management of the trust anchor for the confidentiality protection mechanism is specified with [FMT_MTD.1(CP-AN)].
O.RUNTIME.PROTECTION	Using the runtime protection mechanisms offered to applications is specified by [FPT_FLS.1(FULL), FPT_FLS.1(PARTIAL), FPT_FLS.1(INTEL)].

Table 8: Security objectives for the TOE rationale

6.3.3 Security requirements dependency analysis

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

Security functional requirement	Dependencies	Resolution
FAU_GEN.1	FPT_STM.1	FPT_STM.1
FAU_GEN.2	FAU_GEN.1	FAU_GEN.1
	FIA_UID.1	FIA_UID.1
FAU_SAR.1	FAU_GEN.1	FAU_GEN.1
FAU_SAR.2	FAU_SAR.1	FAU_SAR.1

Security functional requirement	Dependencies	Resolution
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-SSH)	[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.2(NET-IKE)	[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.2(NET-TLS)	[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

Security functional requirement	Dependencies	Resolution
FCS_COP.1(CP)	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1]	FCS_CKM.1(SYM)
	FCS_CKM.4	FCS_CKM.4
FCS_RNG.1(SSL-DFLT)	No dependencies.	
FCS_RNG.1(SSL-FIPS)	No dependencies.	
FCS_RNG.1(DM-INIT)	No dependencies.	
FCS_RNG.1(DM-RUN)	No dependencies.	
FCS_RNG.1(DM-FIPS)	No dependencies.	
FCS_RNG.1(NSS)	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-IPTables)
FDP_IFF.1(NI-IPTables)	FDP_IFC.1	FDP_IFC.2(NI)
	FMT_MSA.3	FMT_MSA.3(NI)
FDP_ITC.2(BA)	[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(BA)
		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.	

Security functional requirement	Dependencies	Resolution
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)
FPT_FLS.1(FULL)	No dependencies.	
FPT_FLS.1(PARTIAL)	No dependencies.	
FPT_FLS.1(INTEL)	No dependencies.	
FPT_STM.1	No dependencies.	
FPT_TDC.1(BA)	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.	
FDP_ACC.2(Namespaces)	FDP_ACF.1	FDP_ACF.1(Namespaces)
FDP_ACC.2(Cgroup)	FDP_ACF.1	FDP_ACC.2(Cgroup)
FDP_ACC.2(SECCOMP)	FDP_ACF.1	FDP_ACF_NA.1(SECCOMP)
FDP_ACF.1(Namespaces)	FDP_ACC.1	FDP_ACC.2(Namespaces)
	FMT_MSA.3	FMT_MSA.3(Namespace-CACP)
FDP_ACF.1(Cgroup)	FDP_ACC.1	FDP_ACC.2(Cgroup)
	FMT_MSA.3	FMT_MSA.3(Cgroup-CACP)
FDP_ACF_NA.1(SECCOMP)	FDP_ACC.1	FDP_ACC.2(SECCOMP)
	FMT_MSA.3	FMT_MSA.3(SECCOMP)
FDP_ETC.2(LC)	[FDP_ACC.1 or FDP_IFC.1]	FDP_ACC.2(Namespaces)
FDP_ITC.2(LC)	[FDP_ACC.1 or FDP_IFC.1]	FDP_ACC.2(Namespaces)
	[FTP_ITC.1 or FTP_TRP.1]	FTP_ITC.1
	FPT_TDC.1	FPT_TDC.1(LC)
FIA_UID.2(LC)	No dependencies.	

Security functional requirement	Dependencies	Resolution
FPT_TDC.1(LC)	No dependencies.	
FMT_MSA.1(Namespaces-CACP)	[FDP_ACC.1 or FDP_IFC.1]	FDP_ACC.2(Namespaces)
	FMT_SMR.1	FMT_SMR.2
	FMT_SMF.1	FMT_SMF.1
FMT_MSA.1(Cgroup-CACP)	[FDP_ACC.1 or FDP_IFC.1]	FDP_ACC.2(Cgroup)
	FMT_SMR.1	FMT_SMR.2
	FMT_SMF.1	FMT_SMF.1
FMT_MSA.1(SECCOMP)	[FDP_ACC.1 or FDP_IFC.1]	FDP_ACC.2(SECCOMP)
	FMT_SMR.1	FMT_SMR.2
	FMT_SMF.1	FMT_SMF.1
FMT_MSA.3(Namespace-CACP)	FMT_MSA.1	FMT_MSA.1(Namespaces-CACP)
	FMT_SMR.1	FMT_SMR.2
FMT_MSA.3(Cgroup-CACP)	FMT_MSA.1	FMT_MSA.1(Namespaces-CACP)
	FMT_SMR.1	FMT_SMR.2
FMT_MSA.3(SECCOMP)	FMT_MSA.1	FMT_MSA.1(SECCOMP)
	FMT_SMR.1	FMT_SMR.2
FMT_MTD.1(LC-COMP)	FMT_SMR.1	FMT_SMR.2
	FMT_SMF.1	FMT_SMF.1
FDP_ACC.2(CP)	FDP_ACF.1	FDP_ACF.1(CP)
FDP_ACF.1(CP)	FDP_ACC.1	FDP_ACC.2(CP)
	FMT_MSA.3	FMT_MSA.3(CP)
FDP_CDP.1(CP)	[FDP_ACC.1 or FDP_IFC.2]	FDP_ACC.2(CP)
FMT_MSA.1(PSO)	[FDP_ACC.1 or FDP_IFC.1]	FDP_ACC.1(PSO)
	FMT_SMR.1	FMT_SMR.2
	FMT_SMF.1	FMT_SMF.1
FMT_MSA.1(TSO)	[FDP_ACC.1 or FDP_IFC.1]	FDP_ACC.1(PSO)
	FMT_SMR.1	FMT_SMR.2
	FMT_SMF.1	FMT_SMF.1

Security functional requirement	Dependencies	Resolution
FMT_MSA.1(CP)	[FDP_ACC.1 or FDP_IFC.1]	FDP_ACC.2(CP)
	FMT_SMR.1	FMT_SMR.2
	FMT_SMF.1	FMT_SMF.1
FMT_MSA.3(PSO)	FMT_MSA.1	FMT_MSA.1(PSO)
	FMT_SMR.1	FMT_SMR.2
FMT_MSA.3(TSO)	FMT_MSA.1	FMT_MSA.1(TSO)
	FMT_SMR.1	FMT_SMR.2
FMT_MSA.3(NI)	FMT_MSA.1	See OSPP rationale.
	FMT_SMR.1	FMT_SMR.2
FMT_MSA.3(CP)	FMT_MSA.1	FMT_MSA.1(CP)
	FMT_SMR.1	FMT_SMR.2
FMT_MSA.4(PSO)	[FDP_ACC.1 or FDP_IFC.1]	FDP_ACC.1(PSO)
FMT_MTD.1(AE)	FMT_SMR.1	FMT_SMR.2
	FMT_SMF.1	FMT_SMF.1
FMT_MTD.1(AS)	FMT_SMR.1	FMT_SMR.2
	FMT_SMF.1	FMT_SMF.1
FMT_MTD.1(AT)	FMT_SMR.1	FMT_SMR.2
	FMT_SMF.1	FMT_SMF.1
FMT_MTD.1(AF)	FMT_SMR.1	FMT_SMR.2
	FMT_SMF.1	FMT_SMF.1
FMT_MTD.1(NI)	FMT_SMR.1	FMT_SMR.2
	FMT_SMF.1	FMT_SMF.1
FMT_MTD.1(IAT)	FMT_SMR.1	FMT_SMR.2
	FMT_SMF.1	FMT_SMF.1
FMT_MTD.1(IAF)	FMT_SMR.1	FMT_SMR.2
	FMT_SMF.1	FMT_SMF.1
FMT_MTD.1(IAU)	FMT_SMR.1	FMT_SMR.2
	FMT_SMF.1	FMT_SMF.1

Security functional requirement	Dependencies	Resolution
FMT_MTD.1(SSH)	FMT_SMR.1	FMT_SMR.2
	FMT_SMF.1	FMT_SMF.1
FMT_MTD.1(SSSD)	FMT_SMR.1	FMT_SMR.2
	FMT_SMF.1	FMT_SMF.1
FMT_MTD.1(SSL)	FMT_SMR.1	FMT_SMR.2
	FMT_SMF.1	FMT_SMF.1
FMT_MTD.1(AM-AP)	FMT_SMR.1	FMT_SMR.2
	FMT_SMF.1	FMT_SMF.1
FMT_MTD.1(AM-MR)	FMT_SMR.1	FMT_SMR.2
	FMT_SMF.1	FMT_SMF.1
FMT_MTD.1(AM-MD)	FMT_SMR.1	FMT_SMR.2
	FMT_SMF.1	FMT_SMF.1
FMT_MTD.1(AM-MA)	FMT_SMR.1	FMT_SMR.2
	FMT_SMF.1	FMT_SMF.1
FMT_MTD.1(CP-AN)	FMT_SMR.1	FMT_SMR.2
	FMT_SMF.1	FMT_SMF.1
FMT_MTD.1(CP-UD)	FMT_SMR.1	FMT_SMR.2
	FMT_SMF.1	FMT_SMF.1
FMT_REV.1(OBJ)	FMT_SMR.1	FMT_SMR.2
FMT_REV.1(USR)	FMT_SMR.1	FMT_SMR.2
FMT_SMF.1	No dependencies.	
FMT_SMR.2	FIA_UID.1	FIA_UID.1
FDP_ETC.2(LS)	[FDP_ACC.1 or FDP_IFC.1]	FDP_IFC.2(LS)
FDP_IFC.2(LS)	FDP_IFF.1	FDP_IFF.2(LS)
FDP_IFF.2(LS)	FDP_IFC.1	FDP_IFC.2(LS)
	FMT_MSA.3	FMT_MSA.3(LS)
FDP_ITC.1(LS)	[FDP_ACC.1 or FDP_IFC.1]	FDP_IFC.2(LS)
	FMT_MSA.3	FMT_MSA.3(LS)

Security functional requirement	Dependencies	Resolution
FDP_ITC.2(LS)	[FDP_ACC.1 or FDP_IFC.1]	FDP_IFC.2(LS)
	[FTP_ITC.1 or FTP_TRP.1]	FTP_ITC.1
	FPT_TDC.1	FPT_TDC.1(LS)
FMT_MSA.1(LS)	[FDP_ACC.1 or FDP_IFC.1]	FDP_IFC.2(LS)
	FMT_SMR.1	FMT_SMR.2
	FMT_SMF.1	FMT_SMF.1
FMT_MSA.3(LS)	FMT_MSA.1	FMT_MSA.1(LS)
	FMT_SMR.1	FMT_SMR.2
FPT_TDC.1(LS)	No dependencies.	

Table 9: TOE SFR dependency analysis

6.4 Security Assurance Requirements

The security assurance requirements for the TOE are the Evaluation Assurance Level 4 components, augmented by ALC_FLR.3, as specified in [CC] part 3. No operations are applied to the assurance components apart from the operation to ASE_CCL.1 as defined in [OSPP].

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

The following table shows the SARs, and the operations performed on the components according to CC part 3: iteration (Iter.), refinement (Ref.), assignment (Ass.) and selection (Sel.).

Security assurance class	Security assurance requirement	Source	Operations			
			Iter.	Ref.	Ass.	Sel.
ASE Security Target evaluation	ASE_CCL.1 Conformance claims	CC Part 3	No	Yes	No	No
	ASE_INT.1 ST introduction	CC Part 3	No	No	No	No
	ASE_SPD.1 Security problem definition	CC Part 3	No	No	No	No
	ASE_OBJ.2 Security objectives	CC Part 3	No	No	No	No
	ASE_ECD.1 Extended components definition	CC Part 3	No	No	No	No
	ASE_REQ.2 Derived security requirements	CC Part 3	No	No	No	No
	ASE_TSS.1 TOE summary specification	CC Part 3	No	No	No	No
ADV Development	ADV_ARC.1 Security architecture description	CC Part 3	No	No	No	No
	ADV_FSP.4 Complete functional specification	CC Part 3	No	No	No	No

Security assurance class	Security assurance requirement	Source	Operations			
			Iter.	Ref.	Ass.	Sel.
	ADV_IMP.1 Implementation representation of the TSF	CC Part 3	No	No	No	No
	ADV_TDS.3 Basic modular design	CC Part 3	No	No	No	No
AGD Guidance documents	AGD_OPE.1 Operational user guidance	CC Part 3	No	No	No	No
	AGD_PRE.1 Preparative procedures	CC Part 3	No	No	No	No
ALC Life-cycle support	ALC_CMC.4 Production support, acceptance procedures and automation	CC Part 3	No	No	No	No
	ALC_CMS.4 Problem tracking CM coverage	CC Part 3	No	No	No	No
	ALC_DEL.1 Delivery procedures	CC Part 3	No	No	No	No
	ALC_DVS.1 Identification of security measures	CC Part 3	No	No	No	No
	ALC_FLR.3 Systematic flaw remediation	CC Part 3	No	No	No	No
	ALC_LCD.1 Developer defined life-cycle model	CC Part 3	No	No	No	No
	ALC_TAT.1 Well-defined development tools	CC Part 3	No	No	No	No
ATE Tests	ATE_COV.2 Analysis of coverage	CC Part 3	No	No	No	No
	ATE_DPT.1 Testing: basic design	CC Part 3	No	No	No	No
	ATE_FUN.1 Functional testing	CC Part 3	No	No	No	No
	ATE_IND.2 Independent testing - sample	CC Part 3	No	No	No	No
AVA Vulnerability assessment	AVA_VAN.3 Focused vulnerability analysis	CC Part 3	No	No	No	No

Table 10: SARs

6.4.1 Security Target evaluation (ASE)

6.4.1.1 Conformance claims (ASE_CCL.1)

Developer action elements:

ASE_CCL.1.1D The developer shall provide a conformance claim.

ASE_CCL.1.2D The developer shall provide a conformance claim rationale.

Content and presentation elements:

ASE_CCL.1.1C The conformance claim shall contain a CC conformance claim that identifies the version of the CC to which the ST and the TOE claim conformance.

ASE_CCL.1.2C The CC conformance claim ~~shall~~*should* describe the conformance of the ST to CC Part 2 as either CC Part 2 conformant or CC Part 2 extended.

ASE_CCL.1.3C The CC conformance claim shall describe the conformance of the ST to CC Part 3 as either CC Part 3 conformant or CC Part 3 extended.

ASE_CCL.1.4C The CC conformance claim shall be consistent with the extended components definition.

ASE_CCL.1.5C The conformance claim shall identify all PPs and security requirement packages to which the ST claims conformance.

ASE_CCL.1.6C The conformance claim shall describe any conformance of the ST to a package as either package-conformant or package-augmented.

ASE_CCL.1.7C The conformance claim rationale shall demonstrate that the TOE type is consistent with the TOE type in the PPs for which conformance is being claimed.

ASE_CCL.1.8C The conformance claim rationale shall demonstrate that the statement of the security problem definition is consistent with the statement of the security problem definition in the PPs for which conformance is being claimed.

ASE_CCL.1.9C The conformance claim rationale shall demonstrate that the statement of security objectives is consistent with the statement of security objectives in the PPs for which conformance is being claimed.

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.

Evaluator action elements:

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

6.5 Security Assurance Requirements Rationale

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

The basis for the justification of EAL4 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

To allow the reader to gain a full understanding of the functionality offered by the TOE and the supporting mechanisms provided by the underlying system the TOE security functionality rests on, the TSS is split into the following parts:

- Discussion of the TOE security functionality
- Documentation of the functionality required from the underlying system to support the TOE security functionality

7.1 Support Mechanisms Offered by the IT Environment

The security functionality implemented by the TOE rests on features offered by the IT environment. This section outlines the mechanisms of the IT environment the TOE utilizes

To implement the concept of a reference monitor, the TOE implements a kernel which operates in a privileged state. The underlying CPU offers such privileged state by restricting numerous processor instructions to be accessible in that privileged state. In addition, a number of processor registers used to alter the behavior of the CPU can only be manipulated in privileged mode. This includes the memory management unit the underlying CPU must offer which can only be configured in privileged mode. As the TOE is capable of running on different CPU architectures, the following listing covers the mentioned functionality for the different CPUs:

- x86 architecture: The kernel executes in ring 0 and user space executes in ring 3. In case the virtualization technique is used, the kernel operates in ring 0 in VM-root mode. User space operates in ring 3 VM-root mode. A virtual machine guest operates in non-VM-root mode and is allowed to use all processor rings.
- POWER: The kernel operates in supervisor mode. User space application execute in problem state. The underlying LPAR hypervisor is assumed to implement a check that hypervisor calls are only accepted from supervisor state and not from problem state (note, the CPU allows the invocation of hypervisor calls from problem state).
- z/Architecture: The kernel operates in supervisor mode. User space application execute in problem state.

In addition to the CPU support for implementing a reference monitor, the TOE uses the following mechanisms from the underlying IT environment to implement security functions:

- NX bit: The No-eXecute bit for memory pages allow the TOE to mark information stored in marked memory pages as non-executable. If the CPU is requested to execute code in those pages, the CPU would raise an exception. The CPU ensures that the NX bit can only be unset by software executing in privileged mode.
- Timer: The underlying platform provides a clock that allows the TOE to maintain a clock. The platform ensures that only software operating in privileged mode is able to alter the clock.

When the TOE executes in a virtual environment, this environment must ensure that the entropy gathering mechanism of the TOE is unaffected by the virtualization support. The entropy gathering rests on the following mechanisms which must behave identical to the TOE executing on native hardware:

- Human Interface Devices (HID): If virtual or real HID are connected to the TOE executing as a guest operating system, the events triggered by the HID must all originate from human users. Under no circumstances shall the host automatically trigger HID events such that the randomness of these HID events provided by humans is non-existent. For example,

key strokes by humans always have a random factor in their timing behavior on which the entropy collection of the TOE rests. The host must not automatically trigger key strokes that lack that human randomness. This requirement is achieved with the allowed virtual environment, because KVM together with QEMU does not implement HID which can be programmed to operate without human intervention.

- Block device events: The virtual environment must ensure that the block device events from platter-based hard disks (i.e. no Solid State Disks) must be immediately relayed to the virtual machine. This means that any kind of buffering present in the host must be disabled for block device events, regardless whether the block device the guest is able to access is backed by a file, a disk partition in the underlying host, a real physical disk or any other means. This implies that, for example, block devices backed by RAM in the host are disallowed in the evaluated configuration of guests. This requirement can be achieved with the allowed virtual environment of KVM and QEMU using the following constraints:
  - Any block devices the TOE "sees" must be backed by real platter-based hard disks. That can be achieved by either providing a file-based disk backend, a disk partition from the host or a full physical disk to the guest.
  - The host must not use its buffer cache for accesses to the block device backends. This can be achieved by instructing QEMU to use the O_DIRECT option with the cache=none with a block device definition.
  - The evaluated configuration of the guest must ensure that for any block device offered by the virtual environment and used by the TOE that does not meet the above mentioned requirements, including the use of block devices not using spinning platters, the corresponding SysFS file "/sys/devices/virtual/block/<DEVICE>/queue/rotational" contains a zero.
- Interrupt events: Interrupt events relayed to the TOE must be based on operations caused by the TOE. The virtual environment must not send arbitrary interrupts to the TOE, especially with a fixed timing. This requirement is achieved with the allowed virtual environment, because KVM together with QEMU does not implement interrupts which can be programmed to operate without being triggered by TOE requests.

7.2 Cryptographic Support Offered by IT Environment

The IT environment provides various cryptographic support mechanisms that are used by the TSF to implement various cryptographic services. The evaluated configuration allows the following cryptographic functions implemented by the IT environment to be used:

- IBM System z CPACF instruction set: The cryptographic libraries of the TOE utilize CPACF, if present, to speed up the cryptographic operations needed for different cryptographic services.

The evaluated configuration allows the mentioned cryptographic support to be used for speeding up cryptographic operations.

7.3 TOE Security Functionality

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

The primary security features of the TOE are:

- Audit
- Cryptographic services

- Packet filter
- Identification and Authentication
- Discretionary Access Control
- Mandatory Access Control
- Security Management
- Runtime Protection mechanisms
- Linux Container (not on POWER architecture)

7.3.1 Audit

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

7.3.1.1 Audit functionality

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

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

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

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

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

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

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

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

The system administrator can select the audited events. Individual files can be configured to be audited by adding them to a watch list that is loaded into the kernel. In addition, audit rules can be specified to generate audit data based on a large number of different attributes, including:

- Subject or user identifiers
- Result of the operation (success/failure)
- Object identity
- Operation performed on an object
- System call number
- SELinux label components

The TOE provides a management application that uses the aforementioned netlink interface. This application is used during boot time to load the audit rules from the configuration file /etc/audit/audit.rules. The audit rules can be modified at runtime of the system.

This security function covers the SFRs of: FAU_GEN.1, FAU_SEL.1, FAU_STG.3, FAU_STG.4, FPT_STM.1.

7.3.1.2 Audit trail

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

- Type: indicates the source of the event, such as SYSCALL, PATH, USER_LOGIN, or LOGIN
- Timestamp: Date and time the audit record was generated
- Audit ID: unique numerical event identifier
- Login ID (“auid”), the user ID of the user authenticated by the system (regardless if the user has changed his real and / or effective user ID afterwards)
- Effective user and group ID: the effective user and group ID of the proces s at the time the audit event was generated
- Success or failure (where appropriate)
- (in MLS mode) SELinux label of the subject that caused the event
- (in MLS mode) SELinux label of the target object
- 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

This information is followed by event specific data. In some cases, such as SYSCALL event records involving file system objects, multiple text lines will be generated for a single event, these all have the same time stamp and audit ID to permit easy correlation.

The audit trail is stored in ASCII text. The TOE provides tools for managing ASCII files that can be used for post-processing of audit data. The application ausearch allows selective extraction of records from the audit trail using defined selection criteria. Using the ausearch, the administrator

is able to select the information he wants to review. The tools allow the specification of a fine-grained search pattern where each information component can be searched for, including combinations of these patterns.

The audit trail is stored in files which are accessible by root only. If the audit trail fills up and reaches a warning threshold the administrator is notified about reaching the configured level. If the audit trail is full, the audit daemon rejects fetching new audit logs from the kernel to store them into a file. The kernel buffer holding audit messages fills up. When the kernel audit message buffer is full, the kernel suspends every subject that triggered an auditable event until the buffer is cleared again. This way, operations causing auditable events are prevented. In addition, the audit daemon can inform the administrator about the full audit trail, can switch to single user mode or halt the system, depending on the configuration.

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

7.3.2 Cryptographic services

The TOE offers different cryptographic services to protect user data. The following subsections cover the different types of cryptographic services analyzed as part of the evaluation. Additional cryptographic mechanisms are active in the TOE which, however, are not subject to the assessments of this evaluation.

7.3.2.1 Cryptographic network services

The TOE provides cryptographically secured network communication channels to allow remote users to interact with the TOE. Using one of the following cryptographically secured network channels, a user can request the following services:

- OpenSSH: The OpenSSH application provides access to the command line interface of the TOE. Users may employ OpenSSH for interactive sessions as well as for non-interactive sessions. 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 application suite 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.
- SSSD: The SSSD information provider connecting to the remote IPA server is able to use TLS v1.1 or TLS v1.2 for protecting the communication link. The userspace component of SSSD handles the key material. The cryptographic primitives are provided by NSS.

In addition to the cryptographically secured communication channels, the TOE also provides cryptographic algorithms for general use.

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

Reference	Description	Implementation Details
[RFC4253] 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.
[RFC4253] section 6.2	Compression	OpenSSH supports the OPTIONAL "zlib" compression method.
[RFC4253] section 6.3	Encryption	The ciphers supported in the evaluated configuration are listed in FCS_COP.1(NET) for the SSH protocol.
[RFC4252] 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.
[RFC4252] 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.
[RFC4252] chapter 8	Password change request and setting new password	The OpenSSH implementation supports the optional password change mechanism in the evaluated configuration.
[RFC4252] chapter 9	Host-Based Authentication: "hostbased"	This OPTIONAL authentication method is disabled in the evaluated configuration.

Table 11: SSH implementation notes

The TOE supports the generation of RSA, DSA and 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 Linux kernel random number generator. 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(DSA), FCS_CKM.1(ECDSA).

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 exchange 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, DSA 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-SSH), FCS_CKM.4, FCS_COP.1(NET), FCS_RNG.1(SSL-DFLT), FCS_RNG.1(SSL-FIPS), FTP_ITC.1, FMT_SMF_RMT.1.

IPSEC and IKEv1 / IKEv2 Protocol Family

The TOE implements the protocol family of IPSEC and IKE with the kernel supported by the Libreswan user application. IPsec can be split into two aspects which are implemented in different locations:

- 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 Linux, 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 might be used to generate and/or sign such a certificate (allowing a wide community trusting this CA to validate the certificate). In a closed community it might also be sufficient to have one server within the community to act as a CA. The NSS library provides the functions to set up such a CA, but those functions are not subject of this Security Target.

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-IKE), FCS_CKM.4, FCS_COP.1(NET), FCS_RNG.1(NSS), FTP_ITC.1, FMT_SMF_RMT.1.

TLS

The TOE provides TLSv1.1 and TLSv1.2 to establish communication to a remote server, i.e. the SFRs claim the TLS client side only. 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 (SSSD IPA client) validates the server certificate.

The TLS protocol within the TOE tunnels the SSSD IPA communication securely between the TOE client and a remote IPA server system.

The following RFCs are supported for implementing the TLS protocol:

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

The following table documents implementation details concerning the NSS 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	NSS 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 Combatibility	The NSS implementation supports the backwards compatible protocol, but this is disabled in the evaluated configuration. It permits use of TLSv1.1 TLSv1.2 exclusively.

Table 12: 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 NSS random number generator. The evaluated configuration also allows the use of an externally-generated certificate. A widely accepted Certification Authority might be used to generate and/or sign such a certificate (allowing a wide community trusting this CA to validate the certificate). In a closed community it might also be sufficient to have one server within the community to act as a CA. The NSS library provides the functions to set up such a CA, but those functions are not subject of this Security Target.

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-TLS), FCS_CKM.4, FCS_COP.1(NET), FCS_RNG.1(NSS), FTP_ITC.1, FMT_SMF_RMT.1.

Confidentiality protected data storage

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

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 now 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 as the file system can now be read. In case data is written to the device, it is transparently encrypted using the master volume key.

Once the dm_crypt protected block device is unlocked and mounted, it is accessible as any other file system. When it is unmounted and locked (i.e. the kernel is informed to discard the master volume key), all data on the block device is inaccessible. Even administrative users like the root user is 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 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 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 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 tool can 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 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), FCS_RNG.1(DM-FIPS), FDP_ACC.2(CP), FDP_ACF.1(CP), FDP_CDP.1, FMT_MSA.1(CP), FMT_MSA.3(CP), FMT_MTD.1(CP-AN), FMT_MTD.1(CP-UD).

7.3.3 Packet filter

The Linux 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 mechanism are provided:

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

7.3.3.1 Network layer filtering

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 possible 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 all of the 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.

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 Linux 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 forwaring 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 applies 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(BA)
- Maintenance of rules: FIA_ATD.1(TU)

7.3.4 Identification and Authentication

User identification and authentication in the TOE includes all forms of interactive login (e.g. using the SSH protocol or log in at the local console) as well as identity changes through the su 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 of authentication.

7.3.4.1 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 to:

- Provides login control and establishing all UIDs, GIDs and login ID for a subject
- Ensure the quality of passwords
- Enforce limits for accounts (such as the number of maximum concurrent sessions allowed for a user)
- Enforce 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
- In MLS mode, it sets up of the sensitivity label and file system name space

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

SSSD is a system daemon with the primary function of providing access to identity and authentication remote resource through a common framework that can provide caching and offline support to the system. It provides PAM and NSS modules. It provides also a better database to store local users as well as extended user data. SSSD can be configured to use a native IPA domain (that is, an IPA identity provider with IPA authentication). One of the primary benefits of SSSD is offline authentication. This solves the case of users having a separate corporate account and a local machine account because of the common requirement to implement a Virtual Private Network (VPN). SSSD can cache remote identities and authentication credentials. This means that a user can still authenticate with these remote identities even when a machine is offline. In an SSSD system, a user only needs to manage one account. SSSD integrates with the PAM and NSS framework and can therefore be used together with PAM modules for local credential stores.

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.

7.3.4.2 User Identity Changing

Users can change their identity (i.e., switch to another identity) using one of the following commands provided with the TOE:

su command

The su command is intended for a switch to a 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. 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, file system and effective 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.

sudo command

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

Sudo is associated with sophisticated ruleset that can be engaged to specify which:

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

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

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.

7.3.4.3 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 for each user the user’s name, the id of the user, an indicator whether the password of the user is valid, the principal group id of the user and other (not security relevant) information. The file /etc/shadow contains for each user a hash of the user's password, the userid, the time the password was last changed, the expiration time as well as the validity period of the password and some other information that are not subject to the security functions as defined in this Security Target. Users are allowed to change their passwords by using the passwd command. This application is able to read and modify the contents of /etc/shadow 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/log/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 with the files /etc/issue for log ins via the physical console or /etc/issue.net for remote log ins, such as via SSH. When performing a log in on the physical console, the banner is displayed above the username and password prompt. For log ins 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.

7.3.4.4 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, DSA, 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 operations performed during 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) for more information.

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

7.3.4.5 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.

7.3.5 Discretionary Access Control

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

A subject 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_CHOWN: 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. DAC is implemented with permission bits and, when specified, ACLs.

The outlined DAC mechanism applies only to named objects which can be used to store or transmit user data. Other named objects are also covered by the DAC mechanism but may be supplemented by further restrictions. These additional restrictions are out of scope for this evaluation. Examples

of objects which are accessible to users 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) and process signals.

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

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

7.3.5.1 Permission bits

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

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

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

7.3.5.2 Access Control Lists (ACLs)

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

- ext4
- XFS
- tmpfs

An ACL entry contains the following information:

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

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

A default ACL is an additional ACL which may be associated with a directory. This default ACL has no effect on the access to this directory. Instead the default ACL is used to initialize the ACL for any file that is created in this directory. If the new file created is a directory it inherits the default

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

7.3.5.3 File system objects

Access to file system objects is generally governed by permission bits. For the above mentioned file system, ACLs are supported.

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.

7.3.5.4 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.

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

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

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

7.3.5.5 at and cron jobs queues

at and cron jobs can only be accessed (read/added/modified/deleted) by the owning user. The TOE maintains at and cron job queues for each user.

The root user can always access every at or cron job queue.

The at or cron jobs are started with the UIDs/GIDs of the creator of the job.

7.3.5.6 print job queues

In the SELinux-enabled mode, the TOE maintains print job queues for each sensitivity label and applies the MLS access control rules when listing or deleting jobs on top of the access restrictions outlined beforehand. A user who has submitted a print job with label A cannot view the print jobs or request the deletion of his print jobs from print queue for label A when he is logged in with label B. The root user is also restricted by the SELinux rule set.

Each print output is generated with the label written on the front and last page as well as header and footer of each page.

7.3.6 Mandatory Access Control

The TOE supports mandatory access control using sensitivity labels automatically attached to processes and objects. This policy is enforced by the SELinux security module and the TOE specific SELinux policy.

The name "mandatory" access control is based on the fact that only authorized administrators can set and modify any labels assigned to objects or subject. Therefore, the access control is not at the discretion of a user like for DAC.

SELinux together together with its policy is used to implement the multi-level system (MLS) policy. The MLS policy is applied to regular users and all resources these users can access. The MLS policy also implements and enforces the role-based access control functionality. This MLS policy also implements and enforces the role-based access control policy.

7.3.6.1 MLS mode: Multi-level security

Sensitivity labels consist of a hierarchical part (the level) and a non-hierarchical set of categories.

The SELinux security module attaches a “sensitivity label” as part of the security context to the objects defined in Security Policy specification. Note: these sensitivity labels are not assigned to virtual machines and their resources since the applied multi-category functionality is more restrictive than the multi-level security.

During login in time, the TOE assigns a SELinux label to the process spawned for the newly logged in user based on the label assigned to the user by the administrator. If the user is assigned to more than one label, the user can choose which label he wants to utilize for the session. The label attached to the new process will be retained during when child processes are created (i.e. during for as well as execution of a file).

Processes are subjects with associated security contexts. When sending signals using the kill system call, the target process behaves like an object.

In addition a process as a subject also has a security context attached. Each process has an effective or “low” sensitivity label (consisting of a hierarchical level and zero or more categories), and a separate “process clearance” or “high” sensitivity label which must dominate the effective label. The effective level is used for all access checks except for processes with the a specific MLS override attribute. Access control is performed based on the sensitivity labels of the process and the object the process interacts with.

When access attempts by a subject onto an object covered by the Discretionary Access Control are performed, the Mandatory Access Control policy is only enforced after the Discretionary Access Control policy allowed the access attempt. In case the Discretionary Access Control policy denies the access attempt, the denial decision is immediately returned to the calling subject.

Attaching the security context to those objects, evaluating the security context in case of access attempts and managing the security context of subjects and objects is performed by functions that SELinux provides for the kernel hooks defined in the LSM framework. The functions at those hooks ensure that all subjects and objects obtain a security context (including a sensitivity label) when they are created in accordance with the rules of the mandatory access control policy.

This security function covers the SFRs of FDP_IFC.2(LS), FDP_IFF.2(LS), FIA_USB.2, FIA_ATD.1(HU).

at and cron jobs queues

The TOE maintains at and cron job queues for each sensitivity label per user and applies the mandatory access control rules when accessing these queues.

Processes spawned by at or cron are assigned the sensitivity label of the creator of the job.

Export/Import of labeled and unlabeled data

The system supports import and export of unlabeled data from/to single level devices. Changes in device level must be performed manually by the administrator and are auditable.

A data archiving tool permits import and export of labeled filesystem data when used by administrators by creating archives that preserve label information.

The TOE IPsec implementation allows assigning labels to network objects and enforcing the mandatory access control policy based on those labels.

The TOE supports the export of labeled data via a multi-level printer. That printer must be connected to the parallel or USB port. However, the printer cannot be connected via an Ethernet connection as this would allow users to directly access the printer, bypassing the TOE-provided print spooler. The print spooler obtains the label of the process causing the printout and ensures that each page contains the label information. In addition, the print spooler ensures that the banner and trailer page of a print job contains the label information of the printout.

This security function covers the SFRs of FDP_ETC.2(LS), FDP_ITC.1(LS), FDP_ITC.2(LS), FDP_TDC.1(LS).

7.3.7 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 configuration 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.

When the MLS mode is active, the configuration files are also protected with SELinux. The protection implements the role-based access control mechanism allowing only dedicated roles to access certain configuration files.

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 and FMT_MTD.1 iterations.

7.3.7.1 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.
- 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-MD), FMT_MTD.1(AM-AP), FMT_MTD.1(AM-MA).

7.3.7.2 MLS mode: Role-based access control

The TOE allows defining roles in the SELinux policy by assigning the domain types to the role to which a user in that role may transition. Each subject has a single active role at all times.

Each user has a set of permitted roles and a default role (both defined by the administrator) and may select an active role using the newrole command from the set of permitted roles. Rules in the policy also define which transitions between roles are allowed. Role transition requests succeed only if the new role is in the set of permitted roles for the current user, and if the policy allows a transition from the current role to the new role.

Administrators can define additional roles using SELinux loadable policy modules defined using the checkmodule, semodule_package, and semodule utilities as documented in the Evaluated Configuration Guide. A role definition consists of a set of permitted role transitions to or from that role, and a set of SELinux domains which correspond to rights associated with the role. Additional roles may be administrative roles with permission to use domains that have specific privileges, including DAC and MAC override capabilities. The policy tools ensure that role definitions may only be removed from the system if the rule is not included in the permitted rule set for any user.

RBAC access checks are performed whenever a subject accesses an object, with the permission based on the subject’s domain, the object’s type, and the operation attempted. The RBAC policy covers all objects covered by the DAC policy. SELinux “allow” rules define the specific access rights to object types for the domains that are associated with the role. Any access attempt from a domain to an object type that is not explicitly permitted by a SELinux “allow” rule is rejected.

Role-based access checks can veto actions that would normally be permitted by DAC or MAC rules, but can never permit something that would be denied according to DAC or MAC rules. Access is permitted only if all applicable policies (DAC, RBAC, and MLS) agree that the access is permitted.

Whenever an operation would result in an illegal SELinux context for a subject or object, for example an invalid combination of role and SELinux user class, the operation will fail and leave the subject and object properties unchanged (for modifying operations), or refuse creation (for creating operations). This ensures that subjects always have exactly one active role.

This security function covers the SFRs of FMT_MTD.1(AM-MR), FMT_SMR.2.

7.3.7.3 Privileges

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

Functionality accessible in user space are applications that can be invoked by users. 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.

7.3.8 Runtime Protection Mechanisms

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

- Adding a guard variable to functions with vulnerable objects which is checked for correctness after a function ends and before the function pointed to by the return address is evaluated. This guard variable is also known as Stack Canary. The variable is a random number which is generated during startup of an application and used throughout the lifetime of the process.
- Marking the runtime memory of all parts of a binary as read-only apart from heap data before the loaded application gains control. This support is enabled for dedicated TSF applications and specifically compiled user applications. Partial protection is also possible which implies that also the .got.plt section is marked read/writable.

On Intel CPUs starting with Ivy Bridge, the CPU feature SMEP is employed which prevents the kernel to execute code located in user space memory.

This security function covers the SFR of FPT_FLS.1(FULL), FPT_FLS.1(PARTIAL), FPT_FLS.1(INTEL)

7.3.9 Linux Container (not on POWER architecture)

Linux Containers is the implementation of user space "virtualization". Various user space processes operate in their own execution environment called a Linux Container which is isolated from other Linux Containers. An application executing in a Linux Container cannot distinguish the execution

environment from a native system. This process cannot interact with processes associated to other Linux Containers. Processes that are not assigned to a Linux Container have the ability to affect the operation of processes inside Linux Containers.

Linux containers are not supported on the POWER CPU architecture.

When comparing Linux Containers to traditional virtualization environments, the following similarities are visible:

- A Linux Container is logically equivalent to a virtual machine.
- The set of processes executing outside a Linux Container are equivalent to processes executing on a virtualization host outside a virtual machine. Those applications have the ability to affect the operation of virtual machines or Linux Containers, respectively.
- Linux Containers are isolated from one another as virtual machines are. The processes inside a Linux Container cannot interact with processes of other Linux Containers.

The key differences between virtual machines and Linux Containers is that Linux Containers only virtualize the user space. All processes of all Linux Containers execute on the very same instance of the Linux kernel. This also implies that unlike virtual machine monitors, Linux Containers do not need special support from the underlying hardware.

To implement Linux Containers different technical mechanisms offered by the Linux kernel are used jointly. The following mechanisms are used -- these mechanisms are detailed in the subsequent sections:

- Linux namespaces implement the isolation support between processes (see section 7.3.9.1).
- Linux control groups, also called "cgroups", implement the resource assignment and resource limitation enforcement for processes (see section 7.3.9.2).
- Limitation of Linux kernel services is enforced by filtering Linux system call. This mechanism is also called seccomp-filter (see section 7.3.9.3).

As the Linux kernel offers no policy enforcement on how these mechanisms are used, a user space framework to manage, configure, and control Linux Containers is provided. This user space framework is called Docker which uses the services implemented by systemd. Docker consists of a set of applications providing a backend registry and administrative interfaces. When a Linux Container is instantiated, Docker instructs systemd to set up the Linux control groups, instantiates the Linux namespaces needed for the Linux container, and potentially sets up the system call filtering mechanism.

The Docker user space framework, however, is not subject to any claims in that ST. The ST claims the functionality that allows user space containers to be created.

7.3.9.1 Linux Namespaces

The purpose of each namespace is to wrap a particular global system resource in an abstraction that makes it appear to the processes within the namespace as if they are part of an isolated instance of the global resource. With the namespace, multiple instances of a resource can be associated with the same set of meta data.

Currently, the following namespaces are used by the TOE:

mount namespaces
: One use of mount namespaces is to create environments that are similar to chroot jails, but more secure and flexible. Mount namespaces can be nested, so that mounts from a master are automatically propagated to its slaves.

UTS namespaces
Allow for each container to have its own hostname and NIS domain name.

IPC namespaces
Each IPC namespace has its own set of System V IPC identifiers and its own POSIX message queue filesystem.

PID namespaces
Isolation of process IDs within a namespace allows for the same PID to be assigned to different processes in different namespaces, thus allowing migration of processes from one host to another without changing its PID. Processes within a PID namespace can only "see" (and therefore signal) PIDs of the same namespace (or PID namespaces nested below). Each PID namespace can have their own init (PID 1) process. The PID 1 process is vital for the entire user space as it acts as the default parent process reaper when a parent of a process terminates.

network namespaces
Each container can have its own network device and its own applications that bind to the per-namespace port number space; suitable routing rules in the host system can direct network packets to the network device associated with a specific container. This allows, e.g., for multiple web servers in different containers to bind to port 80. The network device can only be accessed via the interfaces exported by the kernel through the networking stack. Direct access to the physical device is not given.

A namespace is another indirection when resolving an object for performing operations on that object. The Security Policy Model outlines the different classes of objects namespaces are implemented.

The kernel uses two categories of namespaces which have different rules associated with them. The first category is the set of namespaces covering regular kernel objects. The second category is the user namespace which not only acts as an indirection layer for the kernel object maintaining the process credentials, but also serves as an arbiter of the privileges a process is associated with. The following subsections discuss the different classes of namespaces in detail.

Kernel object namespaces

All object classes except the user namespace listed in Security Policy Model implement a similar approach for an indirection layer to access kernel objects. This section outlines the architecture of the indirection layers. The explanation applies to all different namespaces unless otherwise noted.

A namespace is a data structure which maintains references to objects associated with the respective namespace. For example, a PID namespace maintains a list of process IDs. An UTS namespace simply holds a string with the system naming information. The Security Policy Model outlines the kernel objects maintained for each type of namespace.

A collection of namespaces is maintained where one instance of each type of namespace is referenced. This collection of namespaces is called the NSProxy. During boot, the kernel creates the initial NSProxy which contains the references to the initial namespaces of each type of namespace. At the time of creation of these initial namespaces, no kernel objects of the types maintained by the different namespaces yet exist. The idea now is that when kernel objects of the types outlined in the Security Policy Model are created, they kernel associates them with the namespace the calling subject is associated with. For example, if a process creates a child process, the kernel now looks up the PID namespace the calling process is associated with and associates the new process with the PID namespace of the calling process.

Every process must be associated with exactly one instance of every type of namespace. That association is established by assigning a reference to an NSProxy to each process. In case a new combination of namespaces is to be created, for example by switching to a new mount namespace, a new NSProxy is created with a reference to the existing namespaces and a reference to the new mount namespace.

Processes are the only acting entities in the system. Only processes can perform operations on the kernel objects that are covered by the different types of namespaces, including creation, modification and destruction. Every time a process performs an operation on these objects, the kernel resolves the NSProxy of the calling process. With this NSProxy, the kernel now can obtain a reference to the namespace instance the kernel object shall be found in and tries to resolve this object. If the object cannot be resolved with this namespace, the object is defined to not exist for the calling process.

The following description about the mount namespace example shall enlighten the reader how namespaces are utilized. This example starts at boot time and follows through various operations a process can perform.

Init mount namespace

During boot time, the init mount namespace is created and associated with the init NSProxy. This operation is performed before any user space process exists. The kernel associates that init NSProxy with the very first process it creates. That very first process is filled by the kernel when it starts /sbin/init to perform the userspace initialization. The init process is now associated with the init NSProxy and therefore with the init mount namespace. Baring any special operations on namespaces, any process forked by init are associated with the init NSProxy and the init mount namespace. When the initialization step of mounting all file systems from /etc/fstab is performed, the kernel uses the NSProxy of the process performing the mount operation. As no namespace-specific operations are yet performed, the kernel resolves the init NSProxy and with it the init mount namespace. The mounted file systems are now held and referenced by the init mount namespace.

For any process that tries to perform operations on the mount point, the kernel converts the name of or the file descriptor to the file system object into a kernel internal representation. During that conversion, the mount namespace the mount point is associated with is resolved. Only when this namespace is identical to the mount namespace pointed to by the NSProxy, the operation is performed.

Any process that tries to perform operations on the file system object, the kernel must resolve the file system object from either the path name or the file descriptor. During resolution, the kernel walks the tree of mount points associated with the process.

New mount namespace

A process now may instruct the kernel to a new mount namespace for itself. For this operation, the kernel creates a new NSProxy with a new mount namespace for the process. The kernel also duplicates the references to the tree of mount points to make a private copy for the calling process. When the process now performs an operation on the mount points, including mounting a new file system, the change is reflected in the new mount namespace and in the tree of mount points. As this tree of mount points is not shared with other processes, that modification remains private to the process.

When process with a new mount namespace mounts a new file system, it can access the file system objects on that file system. If another process wanted to access the same file system, it will not be able to do so as the mount point does not exist for his tree of mount points. Thus, the kernel would return an error.

It is possible that a file system is mounted at a different mount point for a process with a new mount namespace than for another process. For example, /dev/sdb3 could be mounted under /mnt/rootns for the process that is yet part of the init mount namespace. The process with a new namespace may mount /dev/sdb3 under /mnt/newns. The file system object of A on that file system is to be accessed via /mnt/rootns/A for the first process and via /mnt/newns/A for the second process. If, however, the first process tries to access /mnt/newns/A, the kernel would access a different file system object, because /mnt/newns is no mount point to /dev/sdb3 for the first process.

A similar approach is made for employing the other namespaces. Every time the kernel needs to access a kernel object, it tries to resolve the calling process NSProxy and the stored namespace reference to resolve the object pointed to by the namespace.

To ensure a smooth transition of operations on kernel objects to handle namespace operations, the following approach is taken by developers: the mentioned namespace functionality is only applicable to an object and operation if the developer added appropriate code to the kernel. If code handling the namespace functionality is missing, e.g. the object is not yet made "namespace-aware", the object is automatically and implicitly assigned to the init namespace.

This security function covers the SFRs of FDP_ACC.2(Namespaces), FDP_ACF.1(Namespaces), FIA_UID.2.

Network Namespace

All Linux namespaces have only relevance local to the operating system with one exception: the networking namespace. The networking namespace allows the isolated management of network devices and TCP/IP stack configuration from the rest of the host system. After creation of the network namespace, no network device is assigned and therefore cannot be managed. Nonetheless, the TCP/IP stack configuration exported via the proc interface can be used to configure the TCP/IP stack behavior.

The configuration changes are private to the namespace and are not visible or enforced outside of the network namespace. To allow a smooth development, each configuration variable must be explicitly "enlightened" within the kernel to be namespace-aware. That means, only when additional functionality surrounding the processing of the variable is added to the kernel to implement the namespace isolation and layer, that variable can be used within a network namespace. For any configuration variable that has no specific code in the kernel making it "namespace-aware", this variable has a global impact to every namespace. However, to alter such configuration variables,

the variable is implicitly defined to belong to the init network namespace. This means that the calling process must possess capabilities that are enforced in the init namespace to alter this variable.

The administrator of the host system (i.e. the administrator in the init namespaces) can decide to assign a network device to the network namespace. When assigning a device to a network namespace, it will be removed from the namespace it is currently part of and is exclusively assigned to the target network namespace. The network namespace with an assigned network device now has exclusive control over that namespace. That means that processes possessing capabilities in that network namespace can perform administrative operation on that network interface like enabling or disabling the network interface, configuring its IP address and any other administrative operation on that network interface. The administrative operations are limited to the operations allowed by the kernel via the networking stack. Thus, direct access of the hardware of the network device is impossible. When re-assigning a network device to a different namespace, the residual information that remains in the network stack only needs to be reset to prevent accidental information leakage.

With an exclusive network device assigned to a namespace, regular networking operation can commence. The IP address assigned to the network interface allows network software associated with that network namespace to communicate with external entities. The IP address allows the kernel to associate any traffic, either ingress or egress, to be associated with the originating network namespace. With this capability, processes associated with different network namespace may use the same listening ports for ingress or source ports for egress traffic.

Separate network namespaces implicitly contain an isolated instance of the localhost network interface. That means, processes assigned to one namespace can only access localhost IP addresses and ports instantiated by processes belonging to the same network namespace.

When assigning an existing process to a different network namespace, existing communication links are retained even when the establishment of such a communication link would be prohibited by the namespace configuration. In such a case, a bridge between the target network namespace and the originating network namespace for the duration of the network session is established. The same applies to file system objects when a process already has a file descriptor and is reassigned to a new mount namespace.

This security function covers the SFRs of FDP_ETC.2(LC), FDP_ITC.2(LC), FPT_TDC.1(LC).

Namespace Administration

When creating a new process without namespace-specific options, the new process inherits the namespace assignments of the mother process. As mentioned above, the first process created by the kernel during boot is assigned to the init namespaces of each namespace type.

A new user namespace can be created by unprivileged processes. Any other namespace type can only be created by a process holding the CAP_SYS_ADMIN capability. During creation of a namespace, at least one process must be assigned to that new namespace. A new namespace can either be created with clone system call flags, or using the unshare system call. The association of an existing process with a namespace can be altered using either the setns system call or by using the /proc/PID/ns directory.

Even though an unprivileged process can create a new user namespace, the user ID visible outside that namespace cannot be altered by an unprivileged process. Using the proc interface, a process with CAP_SYS_ADMIN can assign a new user ID to a process associated with a user namespace that is enforced on any operations requiring a user ID that is not namespace-aware.

This security function covers the SFRs of FMT_MSA.1(Namespaces-CACP), FMT_MSA.3(Namespace-CACP), FMT_MTD.1(LC-COMP).

7.3.9.2 Control Groups

Control Groups provide a mechanism for aggregating/partitioning sets of tasks, and all their future children, into hierarchical groups with specialized behavior.

At any one time there may be multiple active hierarchies of task cgroups. Each hierarchy is a partition of all tasks in the system.

User-level code may create and destroy cgroups by name in an instance of the cgroup virtual file system, specify and query to which cgroup a task is assigned, and list the task PIDs assigned to a cgroup. Those creations and assignments only affect the hierarchy associated with that instance of the cgroup file system.

On their own, the only use for cgroups is for simple job tracking. The intention is that other subsystems hook into the generic cgroup support to provide new attributes for cgroups, such as accounting/limiting the resources which processes in a cgroup can access. Different controllers use the cgroup framework to implement resource tracking and enforcement of policies. These controllers cover:

- Block IO Controller: Two IO control policies are implemented. First one is proportional weight time based division of disk policy. The priorities of the I/O scheduling process can be altered using this policy. The second policy is throttling policy which can be used to specify upper I/O rate limits on devices.
- CPU Accounting Controller: This controller implements CPU accounting usage for groups of processes.
- cpusets controller: This controller allows assignment of CPUs and memory nodes (as compared to memory sizes) to a set of processes.
- Device controller: The device controller allows the specification of a whitelist of devices which are accessible to a group of processes.
- Freezer controller: This controller is used for checkpointing and restarting process groups. When a process group is checkpointed, the operation of all processes of that group is suspended. In the suspended mode, operations on that group can take place such scheduling of resources as needed by the administrator. After completion of this operation, the group of processes is restarted.
- HugeTLB controller: The HugeTLB controller allows to limit the HugeTLB usage per control group and enforces the controller limit during page fault.
- Memory resource controller: The memory controller isolates the memory behavior of a group of tasks from the rest of the system. Process groups can be given a memory range that the collection of all processes of the group together are limited to.
- Network classifier controller: The Network classifier cgroup provides an interface to tag network packets with a class identifier (classid). The Traffic Controller (tc) can be used to assign different priorities to packets from different cgroups. Also, Netfilter (iptables) can use this tag to perform actions on such packets.
- Network priority controller: The Network priority cgroup provides an interface to allow an administrator to dynamically set the priority of network traffic generated by various applications.
- Resource counter controller: The resource counter, is supposed to facilitate the resource management by controllers by providing common information for accounting.

The kernel cgroup framework provides the minimum essential kernel mechanisms required to efficiently implement such control groups. It has minimal impact on the system fast paths, and provides hooks for the specific controller mentioned above.

Multiple hierarchy support is provided to allow for situations where the division of tasks into cgroups is distinctly different for different subsystems - having parallel hierarchies allows each hierarchy to be a natural division of tasks, without having to handle complex combinations of tasks that would be present if several unrelated subsystems needed to be forced into the same tree of cgroups.

At one extreme, each resource controller or subsystem could be in a separate hierarchy; at the other extreme, all subsystems would be attached to the same hierarchy.

The Linux control group framework exports a virtual file system which is accessible to a trusted administrator only. The systemd application provides an API that accesses this virtual file system. The Docker user space framework uses this systemd API to set up the environment for Linux Containers.

This security function covers the SFRs of FDP_ACC.2(Cgroup), FDP_ACF.1(Cgroup), FMT_MSA.1(Cgroup-CACP), FMT_MSA.3(Cgroup-CACP).

7.3.9.3 System Call Filtering

A large number of system calls are exposed to every userland process with many of them going unused for the entire lifetime of the process. As system calls change and mature, bugs are found and eradicated. A certain subset of userland applications benefit by having a reduced set of available system calls. The resulting set reduces the total kernel surface exposed to the application. System call filtering is meant for use with those applications.

Seccomp filtering provides a means for a process to specify a filter for incoming system calls. The filter is expressed as a Berkeley Packet Filter (BPF) program, as with socket filters, except that the data operated on is related to the system call being made: system call number and the system call arguments. This allows for expressive filtering of system calls using a filter program language with a long history of being exposed to userland and a straightforward data set.

Additionally, BPF makes it impossible for users of seccomp to fall prey to time-of-check-time-of-use (TOCTOU) attacks that are common in system call interposition frameworks. BPF programs may not de-reference pointers which constrains all filters to solely evaluating the system call arguments directly.

Using the prctl system call, BPF rules are injected into the kernel where the rules specify:

- System call number
- Optionally system call arguments
- Action to be performed when a rule is matched

A system call filtering rule is matched when the system call number invoked by the process matches one in the rule in the rule set. In addition, if system call arguments are specified in the rule, these arguments must also match.

The rule set is evaluated completely by the kernel for each system call to identify all rules applicable to the call. The rule action with the highest precedence is used for the process. The reason for this approach is that processes can add new rules to an existing rule set without privilege. Considering that the rule with the highest precedence is to be used, processes can therefore only further restrict the ruleset (i.e. have less system calls at their disposal). As the rule set is inherited during fork, a mother process can define limits for a newly created child process that this child process and all its children are bound to.

This security function covers the SFRs of FDP_ACC.2(SECCOMP), FDP_ACF_NA.1(SECCOMP), FMT_MSA.1(SECCOMP), FMT_MSA.3(SECCOMP).

8 Abbreviations, Terminology and References

8.1 Abbreviations

ACL
Access Control List

API
Application Programming Interface

IPA
"Identity, Policy and Audit" management framework which is provided mainly by the FreeIPA software.

KVM
Kernel Virtualized Machine

HTTP
Hypertext Transfer Protocol

SFR
Security Functional Requirement

SSL
Secure Sockets Layer

ST
Security Target

TCP/IP
Transmission Control Protocol / Internet Protocol

TLS
Transport Layer Security

TOE
Target of Evaluation

TSF
TOE Security Functionality

VM
Virtual Machine

VPN
Virtual Private Network

8.2 Terminology

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

Authentication Data
Authentication data is the data used by users or remote entities to authenticate their claimed identity.

Authorized Administrator
This term refers to a user in one of the defined administrative roles of a Linux system. The TOE associates the user with the UID of zero and named "root" with administrative authorities. Effectively, the UID zero is assigned with all Linux capabilities known to the Linux kernel. Every user who is allowed to log on as that root user, or to switch their UID to the root user is considered an authorized administrator. In addition, any user who is able to execute applications which grant one or more Linux capabilities to be used in an unconditional manner is considered an authorized administrator. Note: the process executing on behalf of the root user must possess MLS override attributes to perform management aspects of the Mandatory Access Control Policy.

Category
A category is the non-hierarchical category of the lower MLS label defined with an SELinux label. Note: an SELinux label consists of four parts where the MLS label is one of them. The MLS label in turn is split into a higher and a lower MLS label part.

Classification
A sensitivity label associated with an object.

Clearance
A sensitivity label associated with a subject or user.

DAC
Discretionary Access Control implemented with permission bits and ACLs.

Data
Arbitrary bit sequences on persistent or transient storage media.

Dominate
Sensitivity label A dominates sensitivity label B if the hierarchical level of A is greater than or equal to the hierarchical level of B, and the category set of label A is a proper subset of or equal to the category set of label B. (cf. Incomparable sensitivity labels).

Guest
Software executing within a virtual machine environment. There can be zero or more guests executing concurrently on the host system.

Host
The host system provides the Linux environment that controls and manages the virtual machines. The host provides the execution environment for every virtual machine.

Information
Any data held within a server, including data in transit between systems.

IOMMU
Input / Output Memory Management Unit. This MMU allows the setup of multiple DMA areas for different virtual machines.

KVM
Kernel-based Virtual Machine.

MLS
Multi-level security

Named Object
In Linux, those objects that are covered by access control policies. The list of objects defined as named objects is provided with FDP_ACC.1.

Object
For Linux, objects are defined by FDP_ACC.1.

OSPP
Operating System Protection Profile

OSPP EP
Operating System Protection Profile Extended Package

PAM
Pluggable Authentication Module - the authentication functionality provided with Linux is highly configurable by selecting and combining different modules implementing different aspects of the authentication process.

Product
The term product is used to define software components that comprise the Linux system.

QEMU
The QEMU software component implements the virtual devices and virtual resources for virtual machines. There is one instance of QEMU per virtual machine. The QEMU software component is also identified as the "kvm" application on the host system.

SELinux
Linux kernel LSM module that is able to implement arbitrary security policies. An SELinux policy distributed with the TOE implements multi-level or multi-category security.

Sensitivity Label
The TOE attaches a sensitivity label to each named object. This label consists of a hierarchical sensitivity level and a set of zero or more categories. The policy defines the number and names of the sensitivity levels and categories.

Subject
There are two classes of subjects in Red Hat Enterprise Linux: i) untrusted internal subject - this is a Linux process running on behalf of some user or providing an arbitrary service, running outside of the TSF (for example, with no privileges); ii) trusted internal subject - this is a Linux process running as part of the TSF (for example: service daemons and the process implementing the identification and authentication of users).

Target Of Evaluation (TOE)
The TOE is defined as the Red Hat Enterprise Linux operating system, running and tested on the hardware and firmware specified in this Security Target. The BootPROM firmware as well as the hardware are not part of the TOE.

Technical user
Entity that interacts with the TOE which is not directly controlled by a human, such as applications, remote systems that autonomously or semi-autonomously interact with the TOE.

User
Any individual/person or technical entity (such as a service added by the administrator on top of the TOE) who has a unique user identifier and who interacts with the product.

User Security Attributes
Defined by functional requirement FIA_ATD.1, every user is associated with a number of security attributes which allow the TOE to enforce its security functions on this user. This also includes the user clearance which defines the maximum sensitivity label a user can have access to.

Virtual devices
See virtual resources for a generic explanation. This definition applies also to virtual devices, but with a focus to devices, such as disks, network cards, graphics cards, and similar.

Virtual machine
A virtual machine is an execution environment where the software executing within the virtual machine has access to the processor's user and supervisor state and resources defined by the host system. Resources include the number of processors, RAM size, physical devices, virtualized devices, communication channels to other virtual machines and the host system. For the KVM environment a virtual machine environment is controlled and provided by the Linux kernel hypervisor functionality plus the QEMU application instantiated for each virtual machine.

Virtual machine environment
See virtual machine.

Virtual resources
Virtual resources are resources that either do not physically exist and do not exist in the host system. Virtual resources are implemented by the virtual machine environment and are provided to the respective virtual machine. For example, virtual resources are special exceptions that can be triggered from the virtual machine environment to request services from the host system, such as para-virtualized drivers. Virtual devices can be considered one form of virtual resources.

8.3 References

CC — **Common Criteria for Information Technology Security Evaluation**
Version 3.1R4
Date September 2012
Location http://www.commoncriteriaportal.org/files/ccfiles/CCPART1V3.1R4.pdf
Location http://www.commoncriteriaportal.org/files/ccfiles/CCPART2V3.1R4.pdf
Location http://www.commoncriteriaportal.org/files/ccfiles/CCPART3V3.1R4.pdf

FIPS180-4 — **Secure Hash Standard**
Date March 2012
Location http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf

FIPS197 — **Advanced Encryption Standard**
Date 2001-11-26
Location http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

OSPP — **BSI Operating System Protection Profile**
Version 2.0
Date 2010

OSPP-AM — **BSI OSPP Extended Package - Advanced Management**
Version 2.0
Date 2010

OSPP-LS **BSI OSPP Extended Package - Labeled Security**
Version 2.0
Date 2010

RFC2401 **Security Architecture for the Internet Protocol**
Date November 1998
Location http://tools.ietf.org/html/rfc2401

RFC2402 **IP Authentication Header**
Date November 1998
Location http://tools.ietf.org/html/rfc2402

RFC2406 **IP Encapsulating Security Payload (ESP)**
Date November 1998
Location http://tools.ietf.org/html/rfc2406

RFC2407 **The Internet IP Security Domain of Interpretation for ISAKMP**
Date November 1998
Location http://tools.ietf.org/html/rfc2407

RFC2408 **Internet Security Association and Key Management Protocol (ISAKMP)**
Date November 1998
Location http://tools.ietf.org/html/rfc2408

RFC2409 **The Internet Key Exchange (IKE)**
Date November 1998
Location http://tools.ietf.org/html/rfc2409

RFC3526 **More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)**
Date May 2003
Location http://tools.ietf.org/html/rfc3526

RFC3602 **The AES-CBC Cipher Algorithm and Its Use with IPsec**
Date September 2003
Location http://tools.ietf.org/html/rfc3602

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

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

RFC4301 **Security Architecture for the Internet Protocol**
Date December 2005
Location http://tools.ietf.org/html/rfc4301

RFC4303 **IP Encapsulating Security Payload (ESP)**
Date December 2005
Location http://tools.ietf.org/html/rfc4303

RFC4307 **Cryptographic Algorithms for Use in the Internet Key Exchange Version 2 (IKEv2)**
Author(s) J. Schiller
Date 2005-12-01
Location http://www.ietf.org/rfc/rfc4307.txt

RFC4346 **The Transport Layer Security (TLS) Protocol Version 1.1**
Date April 2006
Location http://tools.ietf.org/html/rfc4346

RFC4419 **Diffie-Hellman Group Exchange for the Secure Shell (SSH) Transport Layer Protocol**
Date March 2006
Location http://tools.ietf.org/html/rfc4419

RFC5114 **Additional Diffie-Hellman Groups for Use with IETF Standards**
Date January 2008
Location http://tools.ietf.org/html/rfc5114

RFC5246 **The Transport Layer Security (TLS) Protocol Version 1.2**
Date August 2008
Location http://tools.ietf.org/html/rfc5246

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

RFC5996 **Internet Key Exchange Protocol Version 2 (IKEv2)**
Date September 2010
Location http://tools.ietf.org/html/rfc5996

RFC6668 **SHA-2 Data Integrity Verification for the Secure Shell (SSH) Transport Layer Protocol**
Date July 2012
Location http://tools.ietf.org/html/rfc6668