Security Target for Ubuntu 18.04 LTS

Version:	**1.0**
Revision:	**1**
Status:	**Released**
Last Update:	**2020-12-02**
Classification:	**Public**

Trademarks

Ubuntu and the Ubuntu logo are trademarks or registered trademarks of Canonical Group Ltd. in the United Kingdom, 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 SUSE Linux Enterprise Server Version 12 Security Target, Copyright © 2016 by SUSE Linux Products GmbH 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
1.0	2020-12-02	Stephan Mueller	Public release of ST

Table of Contents

List of Tables

1 Introduction

1.1 Security Target Identification

Title:	Security Target for Ubuntu 18.04 LTS
Version:	1.0
Revision:	1
Status:	Released
Date:	2020-12-02
Sponsor:	Canonical Group Ltd.
Developer:	Canonical Group Ltd.
Certification Body:	CSEC
Certification ID:	CSEC 2019029
Keywords:	Security Target, Common Criteria, Linux Distribution

1.2 TOE Identification

The TOE is Ubuntu Version 18.04 Long Term Support with the patch level version of 18.04.4.

1.3 TOE Type

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

1.4 TOE Overview

1.4.1 OSPP 2.0 Usage

This ST is derives its security functional requirements from OSPP v2.0 with the extended package on virtualization. A formal compliance claim is not made as this ST specifies EAL2 whereas OSPP v2.0 mandates EAL4. The functional claims of this ST are derived from OSPP v2.0 and the extended package on virtualization.

OSPP v2.0 extends the ASE_CCL.1 by mandating that the ST author shall perform SFR operations in a special way. This ST is written such that these operations are performed in as specified by the OSPP v2.0.

The terminology definition of OSPP v2.0 and the extended package on virtualization is fully applicable to this ST. To aid the reader, the following terminology definition is re-iterated here:

Compartment

The TOE shall allow execution of multiple, separated compartments on a single trusted system. Each compartment can behave like a single platform separated from other compartments. The TOE enforces this separation and controls communication between compartments, as well as communication with external entities, in accordance with a defined policy.

Hardware virtualization utilizes the hardware, mainly the processor support of a hypervisor state, in addition to the supervisor and user states. The hypervisor state is utilized by a component of the TOE to provide an isolated operating space for itself and to provide

operating spaces to other untrusted entities. These untrusted entities can utilize the supervisor and user states of the processor. For this implementation, a compartment is a separated entity capable of executing a standard operating system.

The term compartment in this ST refers to the virtual machines maintained by the KVM / QEMU mechanism part of the TOE. Thus, the term compartment and virtual machine is used interchangeably in this ST.

1.4.2 Configurations defined with this ST

This security target documents the security characteristics of the Ubuntu Linux distribution.

1.4.3 Overview description

Ubuntu 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 derives all functional requirements from the Operating System Protection Profile OSPP v2.0 together with the following extended packages specified for the OSPP:

- Extended package for Virtualization

Ubuntu provides virtualization environment based on the Linux KVM technology. Ubuntu implements the host system for the virtual machine environment and manages the virtual machines. In addition, Ubuntu provides management interfaces to administer the virtual machine environment as well as full auditing of user and administrator operations.

The KVM technology separates the runtime environment of virtual machines from each other. The Linux kernel operates as the hypervisor to the virtual machines but provides a normal computing environment to administrators of the virtual machines. Therefore, the Linux kernel supports the concurrent execution of virtual machines and regular applications. Ubuntu uses the processor virtualization support to ensure that the virtual machines execute close to the native speed of the hardware.

In addition to the separation of the runtime environment, Ubuntu also provides system-inherent separation mechanisms to the resources of virtual machines. This separation ensures that the large software components used for virtualizing and simulating devices executing for each virtual machine cannot interfere with each other. The AppArmor policy also restricts virtual machines to a set of defined resources assigned to the respective virtual machine. Any other resource, including general operating system resources or resources from other users are inaccessible based on the AppArmor restrictions. Using the AppArmor policy, the virtualization and simulation software instances are isolated. The virtual machine management framework uses AppArmor transparently to the administrator. The virtual machine management framework assigns each virtual machine a unique AppArmor label. In addition, each resource dedicated to this virtual machine is assigned the same AppArmor label. The AppArmor policy enforces based on these labels that a virtual machine can only access its own resources. Resources from other virtual machines, the hosting operating system as well as other users are inaccessible.

1.4.4 Allowed Unclaimed Functionality

The TOE implements mechanisms without any security claims specified in this Security Target. This section outlines such mechanism which are allowed to be used in the evaluated configuration. As these listed mechanisms may interfere with the operation of the claimed security functionality, the evaluation ensures that the interference does not weaken any security functionality.

Ubuntu provides userspace virtualization environment called Linux Containers based on the Linux namespace and Linux control groups technology. Ubuntu implements the host system for the Linux Containers and manages these containers. In addition, Ubuntu provides management interfaces to administer the Linux Containers as well as full auditing of user and administrator operations.

The Linux Container technology separates the runtime environment of applications from each other. Resources accessible by applications cannot be shared with other applications. Non-shared resources include PIDs, network, IPC, user IDs, hostname, and mount points. The Linux kernel enforces the separation of these resources but provides other applications. Therefore, the Linux kernel supports the concurrent execution of Linux Containers and regular applications.

1.4.5 Required Hardware and Software

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

- x86 64bit Intel Xeon processors:
    - Supermicro SYS-5018R-WR

IBM System z based on z/Architecture processors:

- IBM z14

Note: the x86_64 system implements the VT-x including the nested page table support (EPT - Intel). The listed Intel x86 machines provide the IOMMU implementation of VT-d.

1.4.6 Intended Method of Use

1.4.6.1 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 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 the TOE.

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.

1.4.6.2 KVM virtual machines

The TOE provides virtual machine environments which allow other operating systems to execute concurrently with the Ubuntu host system. Each virtual machine is represented as a process in the Ubuntu host system and is subject to the standard Linux constraints for processes. The virtualization and simulation logic that supports the operation of a virtual machine executes within the same process of the respective virtual machine.

Treating each virtual machine as a normal process by the Ubuntu host system allows the concurrent execution of standard applications at the same time. Administrative tools are implemented as standard Linux applications. In addition, the computing environment provided by the Ubuntu host system to users logging into that system does not differ from a standard Ubuntu system even while virtual machines execute. Therefore, standard Linux applications and daemons may operate concurrently with virtual machines. To access the Ubuntu host system for managing virtual machines, the TOE provides management access to the libvirtd virtual machine management daemon via OpenSSH. In addition, the virtual machine consoles can be accessed using VNC via an already established OpenSSH channel. The use of OpenSSH for accessing the libvirtd management daemon as well as VNC is encapsulated in client applications which are not part of the TOE, like virsh and virt-manager.

The administration interfaces provided by Ubuntu ensure that the execution environment as well as the resources allocated to a specific virtual machine are separated from other virtual machine instances. Such a transparent setup aids the administrator in keeping a secure configuration for the host system as well as the virtual machines.

Unique AppArmor labels are assigned to each virtual machine and its resources by the virtual machine management software. The AppArmor policy prevents any access request by a virtual machine to a resource if the AppArmor label does not match. In addition, each virtual machine executes with a non-root UID ensuring that the virtual machine operation cannot have an effect on the overall operating system configured on and enforced by the Ubuntu host system.

When using Ubuntu as a host system for virtual machines, normal users are typically not allowed to log on to the system but different UIDs are used to separate different services or virtual machines provided by the system. In such a case, it is assumed that these processes are responsible for the safeguarding of their data. Note: the evaluated configuration permits the use of the host system for regular operation by normal users.

1.4.6.3 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 Ubuntu are assigned unique user identifiers within the single host system that forms the TOE. This user identifier is used together with the attributes 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.

1.4.7 Major Security Features

The primary security features of the TOE are specified as part of the 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

Ubuntu is a general purpose, multi-user, multi-tasking Linux based operating system. It provides a platform for a variety of applications. In addition, virtual machines provide an execution environment for a large number of different operating systems.

The Ubuntu evaluation covers a potentially distributed network of systems running the evaluated versions and configurations of Ubuntu 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 Ubuntu that run in kernel mode plus a set of 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 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 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:

- Ubuntu in the above mentioned version

The TOE and its documentation are supplied on ISO images distributed via the Ubuntu web site. The TOE includes a package holding the additional user and administrator documentation.

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

- Evaluated Configuration Guide published by Canonical 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 on page 11. 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
- Virtualization support, including hypervisor state, shadow page tables support (see section 7.1.7.1), IOMMU support on Intel x86 systems
- 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

The primary security features of the TOE are:

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.

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, 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 (PBKDF2) implemented with the LUKS mechanism, a user-provided passphrase protects the 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 volume key for the block device is decrypted with the passphrase processed by PBKDF2. 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. 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.

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.

Authoritative Access Control

The TOE supports authoritative or mandatory access control based on the following concept:

- To separate virtual machines and their resources at runtime AppArmor rules defined by AppArmor policies are used. The virtual machine resources are labeled to belong to one particular virtual machine by that policy. In addition a virtual machine is awarded a unique label by that policy. The TOE ensures that virtual machines can only access resources bearing the same label.

Virtual machine environments

The TOE implements the host system for virtual machines. It acts as a hypervisor which provides an environment to allow other operating systems execute concurrently.

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.

Additional Functions

The TOE provides many more functions and mechanisms. The evaluation ensures that all these additional functions do not interfere with the above mentioned security mechanisms in the evaluated configuration. The mechanisms given in the following list, however, may interfere with the security functionality of the TOE and should be allowed in the evaluated configuration. Therefore, the evaluation assesses the functionality to verify that the impact on the security functionality at most adds further restrictions as outlined below.

- Linux Container support: The TOE offers userspace virtualization support via Linux Container. That virtualization support shall be allowed to be used such that it does not interfere with the operation of the security functions. The evaluation ensures that the constraints associated with the use of Linux Containers in the evaluated configuration guide has no adverse impact on the security functionality. In addition, the libvirt daemon is allowed to run with the privileges of the root user to allow management of Linux Containers (note, Linux Containers are not referred to by the term containers used in the remainder of this ST).

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

Functions	Exclusion discussion
eCryptFS	eCryptFS are not allowed to be used in the evaluated configuration. The encryption capability provided with this file system is therefore unavailable to any user.
SMACK	The mandatory access control functionality offered by the SMACK LSM is not assessed by the evaluation and disabled in the evaluated configuration.
SELinux	The mandatory access control functionality offered by the SELinux LSM is not assessed by the evaluation and disabled in the evaluated configuration.
SSL / TLS tunnels	The TOE provides the stunnel application which can be used to establish SSL and TLS tunnels with remote peers. This application however was excluded from evaluation assessment.
GSS-API Security Mechanisms	The GSS-API is used to secure the connection between different audit daemons. The security mechanisms used by the GSS-API, however, is not part of the evaluation. Therefore, A.CONNECT applies to the audit-related communication link.

Table 1: Non-evaluated functionalities

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

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.
- 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.
- The default configuration for identification and authentication are the defined password-based PAM modules as well as public-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 subject to 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 are not permitted in the evaluated configuration.

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. The administrator must ensure that the synchronized UIDs to not conflict with the security policy applicable to the TOE.)

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 links between this network and untrusted networks (e. g. the Internet) need to be protected by appropriate measures such as carefully configured firewall systems that prohibit attacks from the untrusted networks. Those protections are part of the TOE environment.

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:
    - Ext3 - standard file system for general data
    - Ext4 - standard file system for general data
    - XFS - standard file system for general data
    - VFAT - special purpose file system for UEFI BIOS support mounted at /boot/efi
    - iso9660 - ISO9660 file system for CD-ROM and DVD
    - tmpfs - the temporary file system backed by RAM
    - rootfs - the virtual root file system used temporarily during system boot
    - procfs - process file system holding information about processes, general statistical data and tunable kernel parameters
    - sysfs - system-related file system covering general information about resources maintained by the kernel including several tunable parameters for these resources
    - devpts - pseudoterminal file system for allocating virtual TTYs on demand
    - devtmpfs - temporary file system that allows the kernel to generate character or block device nodes
    - binfmt_misc - configuration interface allowing the assignment of executable file formats with user space applications
    - securityfs - interface for loadable security modules (LSM) to provide tunables and configuration interfaces to user space
    - cgroup - interface for configuring the control groups mechanism provided by the kernel
    - debugfs - interface for accessing low-level kernel data

  Please note that the TOE supports a number of additional virtual (i.e. without backing of persistent storage) file systems which are only accessible to the TSF - they are not or cannot be mounted. All above mentioned virtual file systems implement access decisions based DAC attributes inferred from the underlying process' DAC attributes. Additional restrictions may apply for specific objects in this file system.
- Virtual machine resources:
    - Disks: files as backing store
    - Disks: device files as backing store

    - Network device: any logical network device present on the host system

  Please note that CPU restrictions and memory range assignments are not listed as part of virtual machine resources, because virtual machines cannot instruct the host system to access these resources. The host system scheduler automatically enforces CPU restrictions. Also, the host system virtual memory management functionality automatically enforces the memory range restriction.
- Inter Process Communication (IPC) objects:
    - Semaphores
    - Shared memory
    - Message queues
    - Named pipes
    - UNIX domain socket special files
- Network sockets (irrespective of their type - such as Internet sockets and netlink sockets)
- Storage device objects (covered by dm_crypt - note that such storage device objects may be provided by either block devices or LVM devices)
- cron 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 session 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
- Any code executed within the virtual machine environment as well as any data stored in resources assigned to virtual machines

2 CC Conformance Claim

This Security Target is CC Part 2 extended and CC Part 3 conformant, with a claimed Evaluation Assurance Level of EAL2, augmented by ALC_FLR.3.

This Security Target does not claim conformance to any Protection Profile.

Common Criteria [CC] version 3.1 revision 5 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.

3.1.1 Assets

Assets to be protected are:

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

3.1.2 Threat Agents

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

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

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

A threat agent might utilize or modify the runtime environment of other compartments in an unauthorized manner.

T.INFOFLOW.COMP

A threat agent might get access to information without authorization by the information flow control policy.

T.COMM.COMP

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.

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,
- 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 public-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.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 and must ensure that only authorized users are able to access such functionality.

O.TRUSTED_CHANNEL

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

O.COMP.INFO_FLOW_CTRL

The TOE will control information flow between compartments under the control of the TOE, based on security attributes of these compartments and potentially other TSF data (e.g., security attributes of objects). This information flow control policy must be able to allow the isolation of individual compartments from other compartments controlled by the TOE.

O.COMP.RESOURCE_ACCESS

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

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.

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.USERDATA T.ACCESS.TSFFUNC

Objective	Threats / OSPs
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.COMM
O.COMP.INFO_FLOW_CTRL	T.INFOFLOW.COMP
O.COMP.RESOURCE_ACCESS	T.ACCESS.COMPENV T.COMM.COMP
O.COMP.IDENT	T.ACCESS.COMPENV T.INFOFLOW.COMP T.COMM.COMP
O.CP.USERDATA	T.ACCESS.CP.USERDATA
O.CP.ANCHOR	P.CP.ANCHOR

Table 2: 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

Objective	Assumptions / Threats / OSPs
OE.INSTALL	A.MANAGE A.DETECT
OE.MAINTENANCE	A.DETECT
OE.PHYSICAL	A.PHYSICAL
OE.RECOVER	A.MANAGE A.DETECT
OE.TRUSTED.IT.SYSTEM	A.PEER.MGT A.PEER.FUNC A.CONNECT
OE.IT.SYSTEM	A.IT.FUNC

Table 3: 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 TSF data controlled by the TOE in transit between trusted IT systems. • O.DISCRETIONARY.ACCESS requiring that data, including TSF data stored with the TOE, have discretionary access control protection. • O.SUBJECT.COM requiring the TSF to mediate communication between subjects.
T.ACCESS.USERDATA	The threat of accessing user data without proper authorization is removed by: • O.CRYPTO.NET requiring cryptographically-protected communication channels for data including user data controlled by the TOE in transit between trusted IT systems. • O.DISCRETIONARY.ACCESS requiring that data including user data stored with the TOE, have discretionary access control protection. • O.SUBJECT.COM requiring the TSF to mediate communication between subjects.
T.ACCESS.TSFFUNC	The threat of accessing TSF functions without proper authorization is removed by:

Threat	Rationale for security objectives
	• O.CRYPTO.NET requiring cryptographically-protected communication channels to limit which TSF functions are accessible to external entities. • O.MANAGE requiring that only authorized users utilize management TSF functions.
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.ACCESS.COMPENV	The threat of utilizing or modifying the runtime environment of compartments executing on behalf of other users is removed by: • O.COMP.RESOURCE_ACCESS requiring the TOE to control access of compartments to objects and resources under its control. • O.COMP.IDENT requiring the TOE to identify the compartment requesting to access resources, objects or information for each access request.
T.INFOFLOW.COMP	The threat of accessing information without authorization by the information flow control policy is removed by:

Threat	Rationale for security objectives
	• O.COMP.INFO_FLOW_CTRL requiring the TOE to control information flow between compartments under the control of the TOE based on security attributes of these compartments and potentially other TSF data. • O.COMP.IDENT requiring the TOE to identify the compartment requesting to access resources, objects or information for each access request.
T.COMM.COMP	The threat of accessing the data communicated between compartments or between a compartment and an external entity is removed by: • O.COMP.RESOURCE_ACCESS requiring the TOE to control access of compartments to objects and resources under its control. • O.COMP.IDENT requiring the TOE to identify the compartment requesting to access resources, objects or information for each access request.
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 4: 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: • 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.

Assumption	Rationale for security objectives
	• OE.INSTALL requiring personnel to ensure that components that comprise the system are distributed, installed and configured in a secure manner supporting the security mechanisms provided by the TOE. • OE.RECOVER requiring personnel to assure that after system failure or other discontinuity, recovery without a protection (security) compromise is achieved.
A.AUTHUSER	The assumption on authorized users to possess the necessary authorization to access at least some of the information managed by the TOE and to act in a cooperating manner in a benign environment is covered by: • OE.ADMIN ensuring that those responsible for the TOE are competent and trustworthy individuals, capable of managing the TOE and the security of the information it contains. • OE.INFO_PROTECT requiring that DAC protections on security-relevant files (such as audit trails and authentication databases) shall always be set up correctly and that users are authorized to access parts of the data maintained by the TOE.
A.TRAINEDUSER	The assumptions on users to be sufficiently trained and trusted to accomplish some task or group of tasks within a secure IT environment by exercising complete control over their user data is covered by: • OE.ADMIN requiring competent personnel managing the TOE. • OE.INFO_PROTECT requiring that those responsible for the TOE must establish and implement procedures to ensure that information is protected in an appropriate manner and that users are trained to exercise control over their own data.
A.DETECT	The assumption that modification or corruption of security-enforcing or security-relevant files will be detected by an administrative user is covered by: • OE.INSTALL requiring an administrative user to ensure that the TOE is distributed, installed and configured in a secure manner supporting the security mechanisms provided by the TOE. • OE.MAINTENANCE requiring an administrative user to ensure that the diagnostics facilities are invoked at every scheduled preventative maintenance period, verifying the correct operation of the TOE. • OE.RECOVER requiring an administrative user to ensure that procedures and/or mechanisms are provided to assure that after system failure or other discontinuity, recovery without a protection (security) compromise is achieved.
A.PEER.MGT	The assumption on all remote trusted IT systems to be under the same management control and operate under security policy constraints compatible with those of the TOE is covered by:

Assumption	Rationale for security objectives
	• 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 5: 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:

OSP	Rationale for security objectives
	• 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.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 6: Sufficiency of objectives enforcing Organizational Security Policies

5 Extended Components Definition

The definition of FCS_RNG has supplied by BSI.

In addition, the Security Target defines the extended component of the FDP_CDP family for usage within this ST.

5.1 Class FDP: User data protection

5.1.1 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.

5.1.1.1 FDP_CDP.1 - Confidentiality for data at rest

Hierarchical to:	No other components.
Dependencies:	[FDP_ACC.1 Subset access control, or FDP_IFC.1 Subset 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 Class FCS: Cryptographic support

5.2.1 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.

5.2.1.1 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 and amended based on discussions with BSI.

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 support functions offered by a subset of the z/Architecture 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.

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 CPU access control policy on instructions as subjects and instructions, 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 CPU access control policy to objects based on the processor state (problem, supervisor, or hypervisor).

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.
- Access to dedicated processor registers is allowed only if the processor is in hypervsior state when the instruction accessing the register is executed.
- CPU instructions restricted to supervisor state are only executed when the CPU is either in supervisor or hypervsior state.
- CPU instructions restricted to hypervsior state are only executed when the CPU is in hypervsior state.

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 or supervisor state.

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 CPU 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 ("problem", "supervisor" or "hypervisor" 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 AES-NI

The AES-NI instruction set is a feature of newer Intel x86 CPUs 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 (AES) (FCS_COP.1(1E))

FCS_COP.1.1
The AES-NI instruction set shall perform encryption and decryption in accordance with a specified cryptographic algorithm AES and cryptographic key sizes 128, 192 or 256 bit that meet the following: FIPS 197, November 6, 2001 (AES).

Application note
AES-NI instruction set consists of several processor instructions implementing aspects of the AES operation. The software must use these different instructions appropriately to implement a full AES operation.

Application note
An Intel Westmere based CPU or newer processor with the instruction set enabled is required in order to use the AES-NI instruction set.

6.1.3 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.

6.1.3.1 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

The operations of assignments and selections are marked with bold font. The operation of refinement is marked with strike through (deletion) or italics (addition). Iterations are marked with an ID added to the SFR number.

The following table shows the SFRs for the TOE, and the operations performed on the components according to CC part 1: 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		CC Part 2	No	No	Yes	No
	FAU_GEN.2 User identity association		CC Part 2	No	No	No	No
	FAU_SAR.1 Audit review		CC Part 2	No	No	Yes	No
	FAU_SAR.2 Restricted audit review		CC Part 2	No	No	No	No
	FAU_SEL.1 Selective audit		CC Part 2	No	No	Yes	Yes
	FAU_STG.1 Protected audit trail storage		CC Part 2	No	No	No	Yes
	FAU_STG.3 Action in case of possible audit data loss		CC Part 2	No	No	Yes	No
	FAU_STG.4 Prevention of audit data loss		CC Part 2	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	No	Yes	No
	FCS_CKM.1(ECDSA) Cryptographic key generation	FCS_CKM.1	CC Part 2	Yes	No	Yes	No
	FCS_CKM.1(EDDSA) Cryptographic key generation	FCS_CKM.1	CC Part 2	Yes	No	Yes	No
	FCS_CKM.2(NET-SSH) Cryptographic key distribution (SSHv2)	FCS_CKM.2	CC Part 2	Yes	Yes	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	Yes	Yes	No
	FCS_COP.1(CP) Cryptographic operation	FCS_COP.1	CC Part 2	Yes	No	Yes	No

Security functional group	Security functional requirement	Base security functional component	Source	Operations			
				Iter.	Ref.	Ass.	Sel.
	FCS_RNG.1(SSL-DFLT) Random number generation (Class DRG.2)	FCS_RNG.1	ECD	Yes	No	Yes	Yes
	FCS_RNG.1(SSL-FIPS) Random number generation (Class DRG.2)	FCS_RNG.1	ECD	Yes	Yes	Yes	Yes
	FCS_RNG.1(DM-DFLT) Random number generation (Class DRG.2)	FCS_RNG.1	ECD	Yes	No	Yes	Yes
	FDP_ACC.1(PSO) Subset access control	FDP_ACC.1	CC Part 2	Yes	No	Yes	No
	FDP_ACC.1(TSO) Subset access control	FDP_ACC.1	CC Part 2	Yes	No	Yes	No
	FDP_ACF.1(PSO) Security attribute based access control	FDP_ACF.1	CC Part 2	Yes	No	Yes	No
	FDP_ACF.1(TSO) Security attribute based access control	FDP_ACF.1	CC Part 2	Yes	No	Yes	No
	FDP_IFC.2(NI) Complete information flow control	FDP_IFC.2	CC Part 2	Yes	No	Yes	No
	FDP_IFF.1(NI-IPTables) Simple security attributes	FDP_IFF.1	CC Part 2	Yes	No	Yes	Yes
	FDP_IFF.1(NI-ebtables) Simple security attributes	FDP_IFF.1	CC Part 2	Yes	No	Yes	Yes
	FDP_ITC.2(BA) Import of user data with security attributes	FDP_ITC.2	CC Part 2	Yes	No	Yes	No
	FDP_RIP.2 Full residual information protection		CC Part 2	No	No	No	Yes
	FIA_AFL.1 Authentication failure handling		CC Part 2	No	Yes	Yes	Yes
	FIA_ATD.1(HU) User attribute definition	FIA_ATD.1	CC Part 2	Yes	Yes	Yes	No
	FIA_ATD.1(TU) User attribute definition	FIA_ATD.1	CC Part 2	Yes	Yes	Yes	No
	FIA_SOS.1 Verification of secrets		CC Part 2	No	No	Yes	No
	FIA_UAU.1 Timing of authentication		CC Part 2	No	No	Yes	No
	FIA_UAU.5 Multiple authentication mechanisms		CC Part 2	No	No	Yes	No

Security functional group	Security functional requirement	Base security functional component	Source	Operations			
				Iter.	Ref.	Ass.	Sel.
	FIA_UAU.7 Protected authentication feedback		CC Part 2	No	No	Yes	No
	FIA_UID.1 Timing of identification		CC Part 2	No	No	Yes	No
	FIA_USB.1 User-subject binding		CC Part 2	No	No	Yes	No
	FPT_STM.1 Reliable time stamps		CC Part 2	No	No	No	No
	FPT_TDC.1(BA) Inter-TSF basic TSF data consistency	FPT_TDC.1	CC Part 2	No	No	Yes	No
	FTA_SSL.1 TSF-initiated session locking		CC Part 2	No	Yes	Yes	No
	FTA_SSL.2 User-initiated locking		CC Part 2	No	Yes	Yes	No
	FTP_ITC.1 Inter-TSF trusted channel		CC Part 2	No	Yes	Yes	Yes
Virtual machine related functionality	FDP_ACC.2(VIRT) Complete access control	FDP_ACC.2	CC Part 2	Yes	No	Yes	No
	FDP_ACF.1(VIRT) Security attribute based access control	FDP_ACF.1	CC Part 2	Yes	No	Yes	No
	FDP_ETC.2(VIRT) Export of user data with security attributes	FDP_ETC.2	CC Part 2	Yes	No	Yes	No
	FDP_IFC.2(VIRT) Complete information flow control	FDP_IFC.2	CC Part 2	Yes	No	Yes	No
	FDP_IFF.1(VIRT) Simple security attributes	FDP_IFF.1	CC Part 2	Yes	Yes	Yes	No
	FDP_ITC.2(VIRT) Import of user data with security attributes	FDP_ITC.2	CC Part 2	Yes	No	Yes	No
	FIA_UID.2(VIRT) User identification before any action	FIA_UID.2	CC Part 2	No	Yes	No	No
	FPT_TDC.1(VIRT) Inter-TSF basic TSF data consistency	FPT_TDC.1	CC Part 2	Yes	No	Yes	No
	FMT_MSA.1(VIRT-CACP) Management of security attributes	FMT_MSA.1	CC Part 2	Yes	No	Yes	Yes
	FMT_MSA.1(VIRT-CIFCP) Management of security attributes	FMT_MSA.1	CC Part 2	Yes	No	Yes	Yes
	FMT_MSA.3(VIRT-CACP) Static attribute initialisation	FMT_MSA.3	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.3(VIRT-CIFCP) Static attribute initialisation	FMT_MSA.3	CC Part 2	Yes	Yes	Yes	Yes
	FMT_MTD.1(VIRT-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	CC Part 2	Yes	No	Yes	Yes
	FMT_MSA.1(TSO) Management of object security attributes	FMT_MSA.1	CC Part 2	Yes	No	Yes	Yes
	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	CC Part 2	Yes	No	Yes	Yes
	FMT_MSA.3(TSO) Static attribute initialisation	FMT_MSA.3	CC Part 2	Yes	No	Yes	Yes
	FMT_MSA.3(NI) Static attribute initialisation	FMT_MSA.3	CC Part 2	Yes	No	Yes	Yes
	FMT_MSA.3(CP) Static attribute initialisation	FMT_MSA.3	CC Part 2	Yes	Yes	Yes	Yes
	FMT_MSA.4(PSO) Security attribute value inheritance	FMT_MSA.4	CC Part 2	No	Yes	Yes	No
	FMT_MTD.1(AE) Management of TSF data	FMT_MTD.1	CC Part 2	Yes	No	Yes	Yes
	FMT_MTD.1(AS) Management of TSF data	FMT_MTD.1	CC Part 2	Yes	No	Yes	Yes
	FMT_MTD.1(AT) Management of TSF data	FMT_MTD.1	CC Part 2	Yes	No	Yes	Yes
	FMT_MTD.1(AF) Management of TSF data	FMT_MTD.1	CC Part 2	Yes	No	Yes	Yes
	FMT_MTD.1(NI) Management of TSF data	FMT_MTD.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_MTD.1(IAT) Management of TSF data	FMT_MTD.1	CC Part 2	Yes	No	Yes	Yes
	FMT_MTD.1(IAF) Management of TSF data	FMT_MTD.1	CC Part 2	Yes	No	Yes	Yes
	FMT_MTD.1(IAU) Management of TSF data	FMT_MTD.1	CC Part 2	Yes	No	Yes	Yes
	FMT_MTD.1(SSH) 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(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	CC Part 2	Yes	Yes	Yes	Yes
	FMT_REV.1(USR) Revocation	FMT_REV.1	CC Part 2	Yes	Yes	Yes	No
	FMT_SMF.1 Specification of management functions		CC Part 2	No	No	Yes	No
	FMT_SMR.1 Security management roles		CC Part 2	No	No	Yes	No

Table 7: 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) **explicit modifications of access rights to objects covered by the access control policies.**

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

a) Date and time of the event, type of event, subject identity (if applicable), and outcome of the event; and

b) For each audit event type, based on the auditable event definitions of the functional components included in the PP/ST, **User identity (if applicable)** .

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 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 command or executing a setuid or setgid program.*

6.2.1.3 Audit review (FAU_SAR.1)

FAU_SAR.1.1 The TSF shall provide **the root user** 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 root user 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).*

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;**
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) **Event type;**
h) **Host identity;**
i) **Virtual machine label;**

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 Unix permission bit settings defined by FDP_ACC.1(PSO) supported by FDP_ACF.1(PSO). The audit trail is readable and writable to the root user only.*

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**.

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 writeable 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

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~~ *the* specified generation ~~algorithm~~ *algorithms:*

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-DFLT) random number generator for use during initialization of confidentiality-protected disks.

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

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: **use of the aforementioned key generation algorithms.**

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

FCS_CKM.1.1 The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm **defined in U.S. NIST FIPS PUB 186-4 appendix B.3.3** 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;*

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

FCS_CKM.1.1 The TSF shall generate 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;*

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

FCS_CKM.1.1 The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm **defined in U.S. NIST FIPS PUB 186-5 (Draft) section 7.4** and specified cryptographic key sizes **defined by the following curves:**

a) Edwards curve 25519;

that meet the following: **U.S. NIST FIPS PUB 186-5 [FIPS186-5DRAFT].**

Application Note:

The TOE supports the generation of EDDSA 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;*

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 ~~a~~ *the* specified cryptographic key distribution ~~method~~ *methods* **as shown below** that ~~meets~~ *meet* the following:

a) Diffie-Hellman key agreement method with diffie-hellman-group14-sha1 defined for the SSH protocol by [RFC4253] supported by [RFC3526];

b) Diffie-Hellman key agreement method with diffie-hellman-group-exchange-sha1 defined for the SSH protocol by [RFC4253] together with [RFC4419];

c) Diffie-Hellman key agreement method with diffie-hellman-group-exchange-sha256 defined for the SSH protocol by [RFC4253] together with [RFC4419];

d) EC Diffie-Hellman key agreement method with ecdh-sha2-nistp256 defined for the SSH protocol by [RFC4253] together with [RFC5656];

e) EC Diffie-Hellman key agreement method with ecdh-sha2-nistp384 defined for the SSH protocol by [RFC4253] together with [RFC5656];

f) EC Diffie-Hellman key agreement method with ecdh-sha2-nistp521 defined for the SSH protocol by [RFC4253] together with [RFC5656];

g) EC Diffie-Hellman key agreement method with curve25519-sha256 defined for the SSH protocol by [SSH25519];

h) 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].

6.2.1.14 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.*

6.2.1.15 Cryptographic operation (FCS_COP.1(NET))

FCS_COP.1.1 The TSF shall perform **encryption, decryption, integrity verification, peer authentication** in accordance with ~~a specified~~ *the following* cryptographic ~~algorithm [assignment: cryptogrphic algorithm]~~ *algorithms,* ~~and~~ cryptographic key sizes ~~[assignment: cryptographic key sizes] that meet the following~~ *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) following FIPS 46-3 and SP800-38A;

2. **AES in CBC mode (aes128-cbc, aes192-cbc, aes256-cbc) following FIPS 197 and SP800-38A;**
3. **AES in CTR mode (aes128-ctr, aes192-ctr, aes256-ctr) following FIPS 197 and SP800-38A;**
4. **AES in GCM mode (aes128-gcm@openssh.com, aes256-gcm@openssh.com) with additional definition in [RFC5647] following FIPS 197 and SP800-38D;**
5. **HMAC with SHA-1 (hmac-sha1-etm@openssh.com) following FIPS 180-4 and FIPS 198-1;**
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] following FIPS 180-4 and FIPS 198-1;**

b) **SSH authentication of host as defined in [RFC4252]:**
1. **RSA signature verification RSASSA-PKCS1-v1.5 using SHA-1 (ssh-rsa) following PKCS #1 v2.2 and FIPS 180-4**
2. **ECDSA with signature verification using SHA-2 (ecdsa-sha2-nistp256 with SHA-256, ecdsa-sha2-nistp384 with SHA-384, ecdsa-sha2-nistp521 with SHA-512) following FIPS 186-4 and FIPS 180-4.**
3. **EdDSA with signature verification using SHA-512 (ssh-ed25519) following [SSH25519] and [FIPS186-5DRAFT] chapter 7.**

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

Application Note: *The SSH protocol allows access to the console of the TOE as well as to the libvirtd virtual machine management daemon.*

6.2.1.16 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) following FIPS 197;**

b) **Block chaining modes: CBC, XTS following SP800-38A and SP800-38E;**

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 using FIPS 180-4.**
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 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 PBKDF2 derives an encryption key from the user's passphrase, the strength of that key directly 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 a reasonable amount of time of several months.*

Application Note: *The master volume key is generated during the "formatting" of the dm-crypt device using the cryptsetup application which obtains the random number that is defined as the master volume key from libgcrypt cryptographic library.*

6.2.1.17 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 **high-resolution time stamps of block device access events, human interface device events and interrupt events as seed source** , the internal state of the RNG shall **have a minentropy of 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 **every time a random number is obtained that is equal in size as the generated random number** generates output for which **2**2**19** strings of bit length 128 are mutually different with probability **of greater than 1-2**-10**.

b) DRG.2.5: Statistical test suites cannot practically distinguish the random numbers from output sequences of an ideal RNG. The random numbers must pass test procedure A.

Application Note:

The OpenSSL library implements the deterministic random number generator usable by mechanisms claimed in this Security Target document.

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

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

a) DRG2.1: If initialized with a random seed using **high-resolution time stamps of block device access events, human interface device events and interrupt events as seed source** , the internal state of the RNG shall **have a minentropy of 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 **every time a random number is obtained that is equal in size as the generated random number** 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: Statistical test suites cannot practically distinguish the random numbers from output sequences of an ideal RNG. The random numbers must pass test procedure A.

Application Note:

The OpenSSL library implements the deterministic random number generator usable by mechanisms claimed in this Security Target document.

6.2.1.19 Random number generation (Class DRG.2) (FCS_RNG.1(DM-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 seed source, Jitter RNG as seed 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 **holding 96 bits of entropy** 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: Statistical test suites cannot practically distinguish the random numbers from output sequences of an ideal RNG. The random numbers must pass test procedure A.

Application Note:

libgcrypt automatically seeds from the getrandom system call, however, this SFR covers the quality of the seed provided with /dev/random.

6.2.1.20 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).

6.2.1.21 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 (includes receive), write (includes send).

6.2.1.22 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: SVTX (sticky bit for directories),

d) Access control security attributes maintained for each partition holding a file system: read-only;

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 written to (which includes modification, creation or removal),

b) A 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 stored in a directory marked with the SVTX bit (sticky 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.

6.2.1.23 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.**

6.2.1.24 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 and receive information mediated by the TOE;

ii. standard Linux processes and processes providing a virtual machine environment that send and receive information mediated by the TOE;

b) Information:

i. Network data routed through the TOE;

ii. No other information;

and all operations that cause that information to flow to and from subjects covered by the SFP.

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

Application Note: *The SFR applies only to host systems which implements the packet filtering functionality.*

6.2.1.25 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) Information security attribute: the logical or physical network interface through which the network data entered the TOE;

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

vi. TCP sequence numbers;

c) Information security attribute: IP packet identified as to be routed by the TSF.

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: **If the packet filter matches the analyzed packet and the rule accepts the packet, the packet is forwarded according to the network protocol stack's behavior**.

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

a) Information security attribute matching;

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;

b) REJECT: then the data is discarded and a notification is returned to the sender.

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

6.2.1.26 Simple security attributes (FDP_IFF.1(NI-ebtables))

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) Information security attribute: the logical or physical network interface through which the network data entered the TOE;

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 IP, TCP, UDP

v. TCP header flags of TOS

vi. Ethernet frames security attributes:

i. Source and destination MAC address,

ii. Ethernet protocol type.

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: **If the packet filter matches the analyzed packet and the rule accepts the packet, the packet is forwarded according to the network protocol stack's behavior**.

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

a) Information security attribute matching;

b) No other concepts;

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

a) Discard the network data without any further processing;

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, and the default rule of the packet filter is DROP then 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.*

6.2.1.27 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 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: **No additional importation control rules.**

6.2.1.28 Full residual information protection (FDP_RIP.2)

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

6.2.1.29 Authentication failure handling (FIA_AFL.1)

FIA_AFL.1.1 The TSF shall detect when **an administrator configurable positive integer within 1 to 1000** 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.

6.2.1.30 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;

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

6.2.1.31 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) Compartments: AppArmor label;

d) Compartments: virtual machine disk resource settings;

e) Compartments: virtual machine network resource settings;

f) Compartments: virtual machine CPU resource settings;

g) Compartments: virtual machine memory resource settings.

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.*

6.2.1.32 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, ECDSA, and EdDSA keys used for the SSH protocol will show the maximum lifetime of a key depending on its size.*

6.2.1.33 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) Local console log-in: banner information;

c) SSH log-in: use of 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.

6.2.1.34 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) No other authentication mechanisms

to support user authentication.

Application Note: *The TOE is able to maintain the following types of software tokens and their verification data:*

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

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) 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.

6.2.1.35 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.

6.2.1.36 Timing of identification (FIA_UID.1)

FIA_UID.1.1 The TSF shall allow

a) Establishing a cryptographically secured network connection;

b) Console log-in: banner information;

c) SSH log-in: use of 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.

6.2.1.37 User-subject binding (FIA_USB.1)

FIA_USB.1.1 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) The user security attributes that are used to enforce the Compartment Access Control Policy.

FIA_USB.1.2 The TSF shall enforce the following rules on the initial association of user 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) Upon instantiating a virtual machine, the label selected by the virtual machine management daemon is associated with the process representing a virtual machine environment.

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 libvirtd virtual machine management daemon automatically identifies a yet unused label during the initial configuration of a virtual machine and stores the information together with the virtual machine configuration in /etc/libvirt/qemu/. This label is applied every time the virtual machine is instantiated.*

FIA_USB.1.3 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_SETUID capability.**

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, they are not listed as a separate mechanism to modify the calling user's UIDs.*

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

Application Note: *The AppArmor label applicable to a virtual machine cannot be modified once it has been applied.*

6.2.1.38 Reliable time stamps (FPT_STM.1)

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

6.2.1.39 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.

6.2.1.40 TSF-initiated session locking (FTA_SSL.1)

FTA_SSL.1.1 The TSF shall lock an interactive *TSF-maintained* session *to a human user* 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-mediated data* 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.

6.2.1.41 User-initiated locking (FTA_SSL.2)

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

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.

6.2.1.42 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:*

- *Cryptographically-protected communication channel using SSH protocol version 2 defined by FCS_CKM.2(NET-SSH) and FCS_COP.1(NET);*

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**.

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 Virtual machine related functionality

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

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

a) Subjects: compartments;

b) Objects:

i) Persistent Storage Objects: all virtual machine resources defined with the Security Policy Model;

ii) Transient Storage Objects: all virtual machine resources defined with the Security Policy Model;

iii) No other objects.

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

Application Note: *Compartments are implemented as processes executing concurrently with standard Linux processes. These compartment processes provide the virtual machine environment.*

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 Security attribute based access control (FDP_ACF.1(VIRT))

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

a) Subject security attributes:

i. AppArmor label;

ii. DAC-based: user ID of the QEMU virtualization process;

iii. **IOMMU-based: the physical resource's DMA address space mapped to the virtual machine;**

b) **Object security attributes:**

i. **AppArmor-based: Disks: file system object name**

ii. **DAC-based: ownership (UID and GID) and DAC permissions of file system object;**

iii. **IOMMU-based: the physical resource's DMA address space;**

iv. **Non-network devices: cgroup device ACLs;**

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) **AppArmor-based: Access of a compartment to an object is allowed when the AppArmor label of the requesting compartment matches the label of the accessed object;**

b) **DAC-based: Access control rules defined with FDP_ACF.1(PSO) and FDP_ACF.1(TSO);**

c) **IOMMU-based: Access of a compartment to a PCI object is granted when the IOMMU of the underlying hardware is configured to link the hardware resource's DMA address space into the virtual machine process address space.**

d) **Non-network devices: The read, write (including ioctl operation triggering object-specific behavior in the kernel) operations of a compartment to a device specified with their type (character/block), major number, minor number is granted if the compartment is assigned to a cgroup that is permitted the respective operation to the device.**

Application Note: *The TOE provides the following types of access control:*

- *If the label of the access-requesting virtual machine matches the label of the accessed resource, access is granted. Otherwise, access is denied to the resource.*
- *DAC: Regular Unix permission bits and ACLs as outlined in the preceding SFRs.*
- *IOMMU-based: The TOE is able to use the IOMMU of the underlying machine to establish the link between the virtual machine process' address space and the hardware resource's DMA address space. Note that the TOE only sets up the memory mapping in the IOMMU. The IOMMU enforces the configuration. Note: the IOMMU functionality supports the PCI device assignment as well as the USB device assignment as the USB controller is a PCI device. In case of USB device assignment, the PCI device of the USB controller is made available via the IOMMU mechanism.*
- *Cgroup device ACLs: If the access-requesting virtual machine matches it is allowed to access the device if the device is assigned to the cgroup the virtual machine operates in.*

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

Application Note: *As the virtual machine processes never execute with root privileges, the DAC override rule for root does not apply here.*

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

6.2.2.3 Export of user data with security attributes (FDP_ETC.2(VIRT))

FDP_ETC.2.1 The TSF shall enforce the **Compartment Access Control Policy and Compartment 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: **The host system ensures that the source IP-address of a virtual machine sending data via network is within the network subnet configured for the bridge between the guest and the host**.

6.2.2.4 Complete information flow control (FDP_IFC.2(VIRT))

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

a) Subjects:
- **i. Compartments;**
- **ii. External entities;**
- **iii. No other entities;**

b) Information:
- **i. User data belonging to compartments;**
- **ii. User data belonging to subjects outside of compartments;**
- **iii. TSF data;**
- **iv. No additional information**

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.

6.2.2.5 Simple security attributes (FDP_IFF.1(VIRT))

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

a) Subject security attributes:
- **i. Process ID of the process providing a virtual machine environment;**
- **ii. No additional attributes;**

b) Information security attributes:
- **i. Any attribute indicating the originating process providing a virtual machine environment;**
- **ii. No TSF data security attributes;**
- **iii. No additional information security attributes.**

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: **No information flow permitted**.

Application Note: *Although virtual machines are implemented using Linux processes, the virtualization mechanism does not allow the use of any Linux inter-process communication by such a process. Therefore, a process cannot communicate with other virtual machines or other Linux processes except via the network channel to the guest system and the VNC network channel to the console of the virtual machine environment. These network channels are mediated by the TOE.*

FDP_IFF.1.3 The TSF shall enforce ~~the~~ **no additional information flow control SFP rules**.

FDP_IFF.1.4 The TSF shall explicitly authorise an information flow based on the following rules: **None**.

FDP_IFF.1.5 The TSF shall explicitly deny an information flow based on the following rules: **None**.

6.2.2.6 Import of user data with security attributes (FDP_ITC.2(VIRT))

FDP_ITC.2.1 The TSF shall enforce the **Compartment Access Control Policy and Compartment Information Flow 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: **The host system ensures that the source IP-address of a virtual machine receiving data via network is within the network subnet configured for the bridge between the guest and the host**.

6.2.2.7 User identification before any action (FIA_UID.2(VIRT))

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

Application Note: *This SFR applies to the process ID of the processes providing a virtual machine environment.*

6.2.2.8 Inter-TSF basic TSF data consistency (FPT_TDC.1(VIRT))

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

FPT_TDC.1.2 The TSF shall use **the IP addresses part of the network packet transmitted by the TSF as specified in RFC 791** when interpreting the TSF data from another trusted IT product.

6.2.2.9 Management of security attributes (FMT_MSA.1(VIRT-CACP))

FMT_MSA.1.1 The TSF shall enforce the **Compartment Access Control Policy** to restrict the ability to **change_default, query, modify, delete** the security attributes **of subjects and objects covered by the SFP, no other security attributes** to **the authorized administrator of the virtual machine configuration**.

Application Note: *The authorized administrator is the user allowed to access the libvirtd daemon and configure virtual machine parameters.*

6.2.2.10 Management of security attributes (FMT_MSA.1(VIRT-CIFCP))

FMT_MSA.1.1 The TSF shall enforce the **Compartment Information Flow Control Policy** to restrict the ability to **change_default, query, modify, delete** the security attributes **of subjects and information covered by the SFP** to **the authorized administrator of the virtual machine configuration**.

Application Note: *The authorized administrator is the user allowed to access the libvirtd daemon and configure virtual machine parameters.*

6.2.2.11 Static attribute initialisation (FMT_MSA.3(VIRT-CACP))

FMT_MSA.3.1 The TSF shall enforce the **Compartment 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 to alter the AppArmor policy files;

b) root user to load the AppArmor policy files into the kernel;

c) root user to alter the object AppArmor label;

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

6.2.2.12 Static attribute initialisation (FMT_MSA.3(VIRT-CIFCP))

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

6.2.2.13 Management of TSF data (FMT_MTD.1(VIRT-COMP))

FMT_MTD.1.1 The TSF shall restrict the ability to **initialize, modify, delete** the **compartment security attributes** to **the authorized administrator of the virtual machine configuration**.

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

Application Note: *The authorized administrator is the user allowed to access the libvirtd daemon and configure virtual machine parameters.*

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** 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** 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: session key used to encrypt and decrypt and data processed on that block device;

d) User security attributes: passphrase that protects the session 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 session 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 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, and ACLs 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** 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 **change_default, modify** 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** 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 writeable 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 writeable 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) as well as FDP_IFF.1(NI-ebtables).*

Application Note: *The iptables and ebtables commands use a netlink interfact 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** 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.

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 writeable 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 network packets by the packet filter;

b) actions performed on the identified network packets by the packet filter;

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

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

Application Note: *The iptables and ebtables commands 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 writeable 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 **modify, delete, initialize** the **user security attributes** 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 writeable to the root user only.*

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 writeable to the root user only.*

6.2.4.18 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 writeable 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.*

6.2.4.19 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.*

6.2.4.20 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**.

6.2.4.21 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.

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).*

6.2.4.22 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.*

6.2.4.23 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 virtual machine configurations;

Application Note: *The given list is kept intentionally generic. 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).*

6.2.4.24 Security management roles (FMT_SMR.1)

FMT_SMR.1.1 The TSF shall maintain the roles:

a) User role with the following rights:

i. Users are authorized to modify their own user password;

ii. Users are authorized to modify the access control permissions for the named objects they own;

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) Role-based access control: set of administrative roles for the role-based access control.

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

Application Note:

Administrative actions can only be performed when the calling subject possesses the above mentioned capabilities which in the TOE configuration is only provided to processes executing with the effective UID or file system UID of zero (also called the root user). As the account for the root user is disabled for direct logon, authorized administrators are defined as users who are assigned to the "sudo" 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.

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

Security functional requirements	Objectives
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(ECDSA)	O.CRYPTO.NET
FCS_CKM.1(EDDSA)	O.CRYPTO.NET
FCS_CKM.2(NET-SSH)	O.CRYPTO.NET
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-DFLT)	O.CP.USERDATA
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_IFF.1(NI-ebtables)	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
FIA_AFL.1	O.I&A
FIA_ATD.1(HU)	O.I&A

Security functional requirements	Objectives
FIA_ATD.1(TU)	O.NETWORK.FLOW
FIA_SOS.1	O.I&A
FIA_UAU.1	O.I&A
FIA_UAU.5	O.I&A
FIA_UAU.7	O.I&A
FIA_UID.1	O.I&A, O.NETWORK.FLOW
FIA_USB.1	O.I&A
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(VIRT)	O.COMP.RESOURCE_ACCESS
FDP_ACF.1(VIRT)	O.COMP.RESOURCE_ACCESS
FDP_ETC.2(VIRT)	O.COMP.INFO_FLOW_CTRL, O.COMP.RESOURCE_ACCESS
FDP_IFC.2(VIRT)	O.COMP.INFO_FLOW_CTRL
FDP_IFF.1(VIRT)	O.COMP.INFO_FLOW_CTRL
FDP_ITC.2(VIRT)	O.COMP.INFO_FLOW_CTRL, O.COMP.RESOURCE_ACCESS
FIA_UID.2(VIRT)	O.COMP.IDENT
FPT_TDC.1(VIRT)	O.COMP.INFO_FLOW_CTRL, O.COMP.RESOURCE_ACCESS
FMT_MSA.1(VIRT-CACP)	O.COMP.RESOURCE_ACCESS
FMT_MSA.1(VIRT-CIFCP)	O.COMP.INFO_FLOW_CTRL
FMT_MSA.3(VIRT-CACP)	O.COMP.RESOURCE_ACCESS
FMT_MSA.3(VIRT-CIFCP)	O.COMP.INFO_FLOW_CTRL
FMT_MTD.1(VIRT-COMP)	O.COMP.INFO_FLOW_CTRL, O.COMP.RESOURCE_ACCESS

Security functional requirements	Objectives
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
FMT_MTD.1(AF)	O.MANAGE
FMT_MTD.1(NI)	O.MANAGE
FMT_MTD.1(IAT)	O.MANAGE
FMT_MTD.1(IAF)	O.MANAGE
FMT_MTD.1(IAU)	O.MANAGE
FMT_MTD.1(SSH)	O.MANAGE
FMT_MTD.1(SSL)	O.MANAGE
FMT_MTD.1(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.1	O.MANAGE

Table 8: 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].
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(ECDSA), FCS_CKM.1(EDDSA)] where the functionality is based on the random number generator as defined by [FCS_RNG.1(SSL-DFLT), FCS_RNG.1(SSL-FIPS)]. 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). 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].
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].
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), FDP_IFF.1(NI-ebtables)]. 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].

Security objectives	Rationale
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].
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.1]. 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] are present.
O.MANAGE	The TOE provides management interfaces globally defined in [FMT_SMF.1] for: • the access control policies [FMT_MSA.1(PSO), FMT_MSA.1(TSO), FMT_MSA.3(PSO), FMT_MSA.3(TSO)]; • the information flow control policy [FMT_MSA.3(NI), FMT_MTD.1(NI)]; • the auditing aspects [FMT_MTD.1(AE), FMT_MTD.1(AS), FMT_MTD.1(AT), FMT_MTD.1(AF)]; • the identification and authentication aspects [FMT_MTD.1(IAT), FMT_MTD.1(IAF), FMT_MTD.1(IAU), FMT_MTD.1(SSH)]. • the session locking threshold [FMT_MTD.1(SSL)]. Persistently stored user data is stored either in hierarchical or relational fashion, which implies an inheritance of security attributes from parent object [FMT_MSA.4(PSO)]. The rights management for the different management aspects is defined with [FMT_SMR.1]. The management interfaces for the revocation of user and object attributes is provided with [FMT_REV.1(OBJ) and FMT_REV.1(USR)].
O.TRUSTED_CHANNEL	The TOE provides a trusted channel protecting communication between a remote trusted IT system and itself [FTP_ITC.1].
O.COMP.INFO_FLOW_CTRL	The information flow control policy covering the runtime of the compartments is specified with [FDP_IFC.2(VIRT)], and [FDP_IFF.1(VIRT)].

Security objectives	Rationale
	As the TOE shall allow export of data belonging to compartments, the TOE assigns the security attributes for enforcing the information flow control policy to the communicated data as specified with [FDP_ETC.2(VIRT)], [FDP_ITC.2(VIRT)], and [FPT_TDC.1(VIRT)]. Management of the security attributes for the information flow control policy is specified with [FMT_MSA.1(VIRT-CIFCP)], and [FMT_MSA.3(VIRT-CIFCP)] as well as FMT_MTD.1(VIRT-COMP).
O.COMP.RESOURCE_ACCESS	The access control policy for the resources belonging to the different compartments is defined with [FDP_ACC.2(VIRT)], [FDP_ACF.1(VIRT)]. 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(VIRT)], [FDP_ITC.2(VIRT)], and [FPT_TDC.1(VIRT)]. Management of the security attributes for the access control policy is specified with [FMT_MSA.1(VIRT-CACP)], and [FMT_MSA.3(VIRT-CACP)] as well as FMT_MTD.1(VIRT-COMP).
O.COMP.IDENT	The identification of compartments to support the information flow control and access control policies is established with [FIA_UID.2(VIRT)].
O.CP.USERDATA	The confidentiality protection mechanism for user data at rest is provided with the access control policy specified with [FDP_ACC.2(CP)] and [FDP_ACF.1(CP)] and supported by the cryptographic operations defined in [FCS_COP.1(CP)]. In addition, the confidentiality mechanism is defined with [FDP_CDP.1(CP)]. The confidentiality protection is implemented with cryptographic mechanisms where the symmetric keys are derived from a random number generator as defined by [FCS_RNG.1(DM-DFLT). The management of the confidentiality protection mechanism is covered by [FMT_MSA.1(CP)] and [FMT_MSA.3(CP)] 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)].

Table 9: Security objectives for the TOE rationale

6.3.3 Security requirements dependency analysis

The following table demonstrates the dependencies of the SFRs modeled in CC Part 2 and the extended component definition in this Security Target, 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
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(ECDSA)	[FCS_CKM.2 or FCS_COP.1]	FCS_COP.1(NET)
	FCS_CKM.4	FCS_CKM.4
FCS_CKM.1(EDDSA)	[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(ECDSA) FCS_CKM.1(EDDSA)
	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(ECDSA) FCS_CKM.1(EDDSA)
	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-DFLT)	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-ebtables)
FDP_IFF.1(NI-IPTables)	FDP_IFC.1	FDP_IFC.2(NI)
	FMT_MSA.3	FMT_MSA.3(NI)
FDP_IFF.1(NI-ebtables)	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) FDP_IFC.2(NI)
	[FTP_ITC.1 or FTP_TRP.1]	FTP_ITC.1
	FPT_TDC.1	FPT_TDC.1(BA)
FDP_RIP.2	No dependencies	
FIA_AFL.1	FIA_UAU.1	FIA_UAU.1
FIA_ATD.1(HU)	No dependencies	
FIA_ATD.1(TU)	No dependencies	
FIA_SOS.1	No dependencies	
FIA_UAU.1	FIA_UID.1	FIA_UID.1
FIA_UAU.5	No dependencies	

Security functional requirement	Dependencies	Resolution
FIA_UAU.7	FIA_UAU.1	FIA_UAU.1
FIA_UID.1	No dependencies	
FIA_USB.1	FIA_ATD.1	FIA_ATD.1(HU)
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(VIRT)	FDP_ACF.1	FDP_ACF.1(VIRT)
FDP_ACF.1(VIRT)	FDP_ACC.1	FDP_ACC.2(VIRT)
	FMT_MSA.3	FMT_MSA.3(VIRT-CACP)
FDP_ETC.2(VIRT)	[FDP_ACC.1 or FDP_IFC.1]	FDP_ACC.2(VIRT) FDP_IFC.2(VIRT)
FDP_IFC.2(VIRT)	FDP_IFF.1	FDP_IFF.1(VIRT)
FDP_IFF.1(VIRT)	FDP_IFC.1	FDP_IFC.2(VIRT)
	FMT_MSA.3	FMT_MSA.3(VIRT-CIFCP)
FDP_ITC.2(VIRT)	[FDP_ACC.1 or FDP_IFC.1]	FDP_ACC.2(VIRT) FDP_IFC.2(VIRT)
	[FTP_ITC.1 or FTP_TRP.1]	FTP_ITC.1
	FPT_TDC.1	FPT_TDC.1(VIRT)
FIA_UID.2(VIRT)	No dependencies	
FPT_TDC.1(VIRT)	No dependencies	
FMT_MSA.1(VIRT-CACP)	[FDP_ACC.1 or FDP_IFC.1]	FDP_ACC.2(VIRT)
	FMT_SMR.1	FMT_SMR.1
	FMT_SMF.1	FMT_SMF.1
FMT_MSA.1(VIRT-CIFCP)	[FDP_ACC.1 or FDP_IFC.1]	FDP_IFC.2(VIRT)
	FMT_SMR.1	FMT_SMR.1
	FMT_SMF.1	FMT_SMF.1

Security functional requirement	Dependencies	Resolution
FMT_MSA.3(VIRT-CACP)	FMT_MSA.1	FMT_MSA.1(VIRT-CACP)
	FMT_SMR.1	FMT_SMR.1
FMT_MSA.3(VIRT-CIFCP)	FMT_MSA.1	FMT_MSA.1(VIRT-CIFCP)
	FMT_SMR.1	FMT_SMR.1
FMT_MTD.1(VIRT-COMP)	FMT_SMR.1	FMT_SMR.1
	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.1]	FDP_ACC.2(CP)
FMT_MSA.1(PSO)	[FDP_ACC.1 or FDP_IFC.1]	FDP_ACC.1(PSO)
	FMT_SMR.1	FMT_SMR.1
	FMT_SMF.1	FMT_SMF.1
FMT_MSA.1(TSO)	[FDP_ACC.1 or FDP_IFC.1]	FDP_ACC.1(PSO)
	FMT_SMR.1	FMT_SMR.1
	FMT_SMF.1	FMT_SMF.1
FMT_MSA.1(CP)	[FDP_ACC.1 or FDP_IFC.1]	FDP_ACC.2(CP)
	FMT_SMR.1	FMT_SMR.1
	FMT_SMF.1	FMT_SMF.1
FMT_MSA.3(PSO)	FMT_MSA.1	FMT_MSA.1(PSO)
	FMT_SMR.1	FMT_SMR.1
FMT_MSA.3(TSO)	FMT_MSA.1	FMT_MSA.1(TSO)
	FMT_SMR.1	FMT_SMR.1

Security functional requirement	Dependencies	Resolution
FMT_MSA.3(NI)	FMT_MSA.1	FMT_MTD.1(NI) is specified to require the management of security attributes for the Network Information Flow Control Policy, just as a potential FMT_MSA.1(NI) would have been specified. However, the Network Information Flow Control Policy is not required to be enforced when managing the security attributes, as the management aspect of the network information flow control functionality is not protected by the network information flow control mechanism. Therefore, FMT_MSA.1 is not applicable and is replaced with FMT_MTD.1(NI).
	FMT_SMR.1	FMT_SMR.1
FMT_MSA.3(CP)	FMT_MSA.1	FMT_MSA.1(CP)
	FMT_SMR.1	FMT_SMR.1
FMT_MSA.4(PSO)	[FDP_ACC.1 or FDP_IFC.1]	FDP_ACC.1(PSO)
FMT_MTD.1(AE)	FMT_SMR.1	FMT_SMR.1
	FMT_SMF.1	FMT_SMF.1
FMT_MTD.1(AS)	FMT_SMR.1	FMT_SMR.1
	FMT_SMF.1	FMT_SMF.1
FMT_MTD.1(AT)	FMT_SMR.1	FMT_SMR.1
	FMT_SMF.1	FMT_SMF.1
FMT_MTD.1(AF)	FMT_SMR.1	FMT_SMR.1
	FMT_SMF.1	FMT_SMF.1
FMT_MTD.1(NI)	FMT_SMR.1	FMT_SMR.1
	FMT_SMF.1	FMT_SMF.1
FMT_MTD.1(IAT)	FMT_SMR.1	FMT_SMR.1
	FMT_SMF.1	FMT_SMF.1
FMT_MTD.1(IAF)	FMT_SMR.1	FMT_SMR.1
	FMT_SMF.1	FMT_SMF.1

Security functional requirement	Dependencies	Resolution
FMT_MTD.1(IAU)	FMT_SMR.1	FMT_SMR.1
	FMT_SMF.1	FMT_SMF.1
FMT_MTD.1(SSH)	FMT_SMR.1	FMT_SMR.1
	FMT_SMF.1	FMT_SMF.1
FMT_MTD.1(SSL)	FMT_SMR.1	FMT_SMR.1
	FMT_SMF.1	FMT_SMF.1
FMT_MTD.1(CP-AN)	FMT_SMR.1	FMT_SMR.1
	FMT_SMF.1	FMT_SMF.1
FMT_MTD.1(CP-UD)	FMT_SMR.1	FMT_SMR.1
	FMT_SMF.1	FMT_SMF.1
FMT_REV.1(OBJ)	FMT_SMR.1	FMT_SMR.1
FMT_REV.1(USR)	FMT_SMR.1	FMT_SMR.1
FMT_SMF.1	No dependencies	
FMT_SMR.1	FIA_UID.1	FIA_UID.1

Table 10: TOE SFR dependency analysis

6.4 Security Assurance Requirements

The security assurance requirements for the TOE are the Evaluation Assurance Level 2 components, augmented by ALC_FLR.3, as specified in [CC] part 3. No operations are applied to the assurance components.

The security assurance requirements (SARs) for the TOE are defined in [CC] part 3 for the Evaluation Assurance Level 2, 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.
ADV Development	ADV_ARC.1 Security architecture description	CC Part 3	No	No	No	No
	ADV_FSP.2 Security-enforcing functional specification	CC Part 3	No	No	No	No
	ADV_TDS.1 Basic design	CC Part 3	No	No	No	No

Security assurance class	Security assurance requirement	Source	Operations			
			Iter.	Ref.	Ass.	Sel.
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.2 Use of a CM system	CC Part 3	No	No	No	No
	ALC_CMS.2 Parts of the TOE CM coverage	CC Part 3	No	No	No	No
	ALC_DEL.1 Delivery procedures	CC Part 3	No	No	No	No
	ALC_FLR.3 Systematic flaw remediation	CC Part 3	No	No	No	No
ASE Security Target evaluation	ASE_INT.1 ST introduction	CC Part 3	No	No	No	No
	ASE_CCL.1 Conformance claims	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
ATE Tests	ATE_COV.1 Evidence of coverage	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.2 Vulnerability analysis	CC Part 3	No	No	No	No

Table 11: SARs

6.5 Security Assurance Requirements Rationale

The basis for the justification of EAL2 augmented with ALC_FLR.3 is the threat environment experienced by the typical consumers of the TOE. This matches the package description for EAL2 (basic).

7 TOE Summary Specification

7.1 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
- Authoritative Access Control
- Virtual machine environments
- Security Management

7.1.1 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.

7.1.1.1 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, all processes are stopped that generate audit records, or the audit daemon

executes an administrator-specified notification action depending on the configuration of the audit daemon. This ensures that audit records do not get lost due to resource shortage and the administrator can backup and clear the audit trail to free disk space for new audit logs.

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

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

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

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

This security function covers the SFRs of: FAU_GEN.1, FAU_GEN.2, FAU_SAR.2, FAU_SEL.1, FAU_STG.1, FAU_STG.3, FAU_STG.4.

7.1.1.2 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 ID: the effective user ID of the process at the time the audit event was generated
- Success or failure (where appropriate)
- Process ID of the subject that caused the event (PID)

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

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

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

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

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

7.1.2 Cryptographic 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.
- libvirtd: The libvirtd daemon is the management facility to allow remote users to configure virtual machines. The configuration covers all aspects such as assigning of resources, starting or stopping of virtual machines. libvirtd directly interacts with the virtual machines. This interface is protected using OpenSSH.
- VNC: The VNC interface provides the access mechanism for users to interact with the console of a virtual machine. The VNC connection is tunneled through OpenSSH.

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

The cryptographic primitives for implementing the above mentioned cryptographic communication protocols are provided by OpenSSL.

7.1.2.1 SSHv2 Protocol

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

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

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

Reference	Description	Implementation Details
RFC 4252 chapter 9	Host-Based Authentication: "hostbased"	This OPTIONAL authentication method is disabled in the evaluated configuration.

Table 12: SSH implementation notes

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

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

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

- Establishing a secure communication channel using the following cryptographic functions provided by the SSH v2.0 protocol:
  - Encryption as defined in section 6.3 of [RFC4253] - the keys are generated using the random number generator of the underlying cryptographic library;
  - Diffie-Hellman key agreement as defined in chapter 8 of [RFC4253];
  - The keyed hash function for integrity protection as defined in section 6.4 of [RFC4253].

  Note: The protocol supports more cryptographic algorithms than the ones listed above. Those other algorithms are not covered by this evaluation and should be disabled or not used when running the evaluated configuration.
- Performing user authentication using the standard password-based authentication method the TOE provides for users (password authentication method as defined in chapter 5 of [RFC4252]).
- Performing user authentication using a RSA, ECDSA, or EdDSA 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 (either the default RNG or an SP800-90A compliant DRBG with AES-256 core with derivation function, without prediction resistance) seeded by /dev/random to generate cryptographic keys. OpenSSL provides different DRNGs depending whether the FIPS 140-2 mode is enabled in the system.

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

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), FMT_SMF.1, FTP_ITC.1.

7.1.2.2 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 volume key which is injected into the kernel. Using that volume key, the kernel is now able to decrypt (to unlock) the data on the device and provides access to data stored on that device. At this point, the file system can be mounted as the file system can now be read.

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

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

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

1. obtain the user's passphrase
2. apply the LUKS key derivation mechanism on the passphrase
3. read the encrypted volume key from the device
4. decrypt the volume key with the key derived from the user's passphrase
5. inject the decrypted volume key into the kernel and set up the mapping between the block device and the volume key

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

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

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

This security function covers the SFRs of FCS_CKM.1(SYM), FCS_RNG.1(DM-DFLT), FCS_COP.1(CP), 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).

7.1.3 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:

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

7.1.3.1 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.

7.1.3.2 Link layer filtering

ebtables chains

Similarly to the netfilter hooks in the network layer handling protocol, the link layer code handling the bridging functionality implements chains that are used by ebtables to apply filtering.

The netfilter framework is used to implement the ebtables chains at the following points in the packet traversal logic:

- When the packet enters the link layer of the TOE and after applying some sanity checks, but before the bridging decision made, the BROUTING chain is triggered.
- After passing the BROUTING chain but still before the bridging decision, the PREROUTING chain is triggered. When the bridging code decides that the frame is not intended for a bridge, it is forwarded to the network layer and processing on the link layer stops.

- After passing the bridge routing decision and the routing code marks the packet to be targeted for another host, the FORWARD chain is triggered.
- After passing the bridge routing decision and the routing code marks the packet to be targeted for the local system, the INPUT chain is triggered.
- When a packet is generated locally and the bridging decision marks the frame to be intended for a bridge, the OUTPUT chain is triggered.
- When the packet traversed all of the bridge logic and is about to be placed on the wire again, the POSTROUTING chain is triggered.

ebtables filtering rules

The packet selection system called ebtables uses the netfilter framework to implement the actual packet filtering logic for Ethernet frames routed through bridges.

Bridged networking communication is only visible on the link layer. As the host system uses bridges to connect virtual machines to the environment, only ebtables can be used to control the traffic flow. The data encapsulated within the Ethernet frames are not routed through the host system's network stack and can therefore not be analyzed and controlled by the IPTables framework outlined above.

Even though the network stack of the host system does not analyze the bridged Ethernet frames any more, the ebtables framework receives the entire frame and is allowed to operate on that frame. Therefore, ebtables can analyze the protocol inside the Ethernet frame.

Please note that the ST author is perfectly aware that the term "packet" is used for the network layer (which includes the TCP/IP protocol family), whereas the term "frame" applies to the link layer (which includes the Ethernet frames). However, when speaking about ebtables that can analyze IP packets inside the link layer, this ST refers to them as packets/frames.

The ebtables framework can apply filters based on the following concepts:

- Matching of the frame/packet: Matches are based on Ethernet protocols or source and/or destination MAC addresses.
- Action to be performed on matched frames: Similarly to IPTables, ebtables must be configured to perform an action on the matched frame. The action can either be to discard the packet or to accept it.

This security function covers the SFRs of:

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

7.1.4 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.

7.1.4.1 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

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.

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.

User accounts are stored in configuration files (/etc/passwd and /etc/shadow). Both are writable to the root user only. In addition, /etc/shadow is readable to the root user only. Modification of both files is performed using a set of administrative applications. When a user ID is removed, its entry in the configuration files is removed by the administrative interfaces. Therefore, a log-in using such a removed user ID is unsuccessful.

This security function covers the SFRs of FIA_AFL.1, FIA_SOS.1, FIA_UAU.1, FIA_UID.1, FIA_UAU.5, FIA_UAU.7, FIA_USB.1, FMT_REV.1(USR).

7.1.4.2 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 SFR of FIA_USB.1.

7.1.4.3 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/tallylog 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(HU).

7.1.4.4 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, ECDSA, or EdDSA 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.

7.1.4.5 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.

7.1.5 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_FOWNER: Setting permissions and ownership on objects even if the process' UID does not match the UID of the object.
- CAP_FSETID: Don't clear SUID and SGID permission bits when a file is modified.

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

7.1.5.1 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). The SVTX (sticky) attribute is used for world-writeable temp directories preventing the removal of files by users other than the owner.

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

When using the VFAT file system at the /boot/efi mount point, the permissions applicable for every file system object beneath that mount point is specified with mount options. The mount system call allows the specification of the owning ID, group ID and the permissions applicable to the file system objects. The permissions specified at the mount point have the same semantics as the Unix permission bits.

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

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

7.1.5.2 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:

- ext3
- 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.

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

7.1.5.3 File system objects

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.

7.1.5.4 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.

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

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

7.1.5.5 cron jobs queues

cron jobs can only be accessed (read/added/modified/deleted) by the owning user. The TOE maintains cron job queues for each user.

The root user can always access every cron job queue.

The cron jobs are started with the UIDs/GIDs of the creator of the job.

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

7.1.6 Authoritative Access Control

The TOE supports authoritative access control using different mechanisms under the sole control of the administrator.

The name "authoritative" 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.

7.1.6.1 Resource access control for virtual machines

The TOE implements the following types of access control restrictions to limit virtual machines to access only their resources:

- AppArmor-based: each virtual machine and its resource is assigned to a unique AppArmor label which prevents other virtual machines with different labels to access either the virtual machine process or its resources.
- IOMMU-based: The TOE is able to configure the IOMMU of the underlying machine to bind the DMA address space of a particular physical resource to be only visible in the address space of the virtual machine process configured to access that resource.
- Cgroup-based: each virtual machine is granted access to a white list of device files. Access to other device files is prevented using the cgroup device ACL mechanism.

The AppArmor mechanism is implemented as an LSM that is invoked for each and every operation of a subject on an object or resource. The AppArmor policy enforced for virtual machines therefore prevent virtual machines from accessing:

- Resources assigned to other virtual machines,
- Resources owned by other users on the system,
- Resources belonging to the host operating system.

This security function covers the SFRs of: FDP_ACC.2(VIRT), FDP_ACF.1(VIRT).

7.1.7 Virtual machine environments

KVM is implemented as part of the Linux kernel supported by user space code. It consists of two essential components that implement VMM functionality: the KVM Linux kernel module and QEMU for hardware emulation. The use of QEMU implies that KVM provides full virtualization to its guests and can, therefore, execute unaltered guest operating systems.

The KVM Linux kernel module implements memory management and virtual machine maintenance functionality. This kernel extension makes the entire Linux kernel the hypervisor. Virtual machines are treated by the Linux kernel as normal applications. The kernel schedules them like applications, and they can be handled like applications. As such, the process implementing a virtual machine can be seen in process listings and it can be sent signals, like SIGTERM.

From the Linux kernel perspective, the virtual machine is just another process. However, the virtual machine process has a special layout. The process image is split into two parts. The first part hosts a regular application logic executing in user mode – this is used to maintain the QEMU I/O virtualization and some other small KVM-related software components. The second part contains the image of the guest code, usually an operating system, where the software may execute either in supervisor or user mode of the processor. This implies that the entire memory used for the guest operating system is allocated by the QEMU application. The kernel keeps track of which parts of the application belong to the guest operating system and which parts to the regular application.

When the kernel releases control of the CPU to the virtual machine process, it sets the processor state of the CPU to the user state when calling the regular application logic in user mode. However, when returning control of the CPU to the guest code, the CPU can be set either to supervisor state or user state, depending on the state of the CPU when the Linux kernel initially obtained control.

The overall logic flow that connects the Linux kernel with the regular application logic and the guest operating system. This logic flow is an endless loop, which can be characterized as follows:

1. The regular application logic executing in user mode sets up the virtual machine configuration by instructing the kernel to allocate memory for the application, CPU and other resources. The kernel sets up these resources and assigns them to the calling process. After setup is complete, the kernel is instructed to execute the guest. This phase starts the loop and is not executed again during the loop.
2. The kernel now causes the processor to enter guest mode. If the processor exits guest mode due to an event such as an external interrupt or a shadow page table fault (see section 7.1.7.1), the kernel performs the necessary handling and resumes guest execution. If the exit is due to an I/O instruction or a signal queued to the process, then the kernel exits to the regular application logic in user mode.
3. The processor executes guest code until it encounters an instruction that needs assistance, a fault, or an external interrupt. The processor then returns control to the host kernel.

4. If the host kernel detects an exit of the guest code due to an I/O instruction or a signal, or until an external event such as arrival of a network packet or a timeout occurs, the kernel invokes the user mode component of the virtual machine process. The processed I/O instructions cover programmed I/O (PIO) whose implementation is not as complex as the second set of processed I/O instructions, the memory mapped I/O (MMIO). QEMU and a small extension for making QEMU KVM-aware is used to implement the I/O handling as it implements a number of emulated devices and mediates access to real resources when a device is accessed via the I/O instruction. QEMU may alter the virtual processor state to reflect the emulated I/O instruction result to the calling guest code. The modification of the virtual processor state is one with IOCTLs to a file descriptor that QEMU has allocated when setting up the virtual machine. This file descriptor is used to store all virtual machine and virtual CPU data relevant for executing the virtual machine. As the file descriptor is bound to one process only, the kernel implicitly ensures that a QEMU instance can only operate on its virtual machine and virtual CPUs. Once QEMU completes the I/O operation, it signals the kernel that the guest code can resume execution which is implemented with step 2 above.

In this architecture, regular applications (i.e., applications executing only in user state of the CPU and having full access to all services of the Linux kernel and, therefore, other parts of the operating system) coexist with applications that host virtual machines.

The libvirtd management daemon sets up virtual machines and controls the resources assigned to virtual machines. To support the separation of virtual machines, libvirtd uses the following capabilities:

- Every virtual machine process executes with the normal, unprivileged user ID of “qemu” and the group ID of “qemu”. This implies that these processes do not possess any Linux kernel capability.
- Libvirtd sets up the unique AppArmor label for a virtual machine and assigns it to the virtual machine and its resources. The resource access control functionality defined as an independent security functionality.
- Libvirtd can instruct the kernel to set up the IOMMU in a way to exclusively assign hardware resources to a virtual machine process as an independent security functionality.
- Every virtual machine process will be placed in a dedicated cgroup. Cgroup is a mechanism of the Linux kernel to mark processes and assign certain properties to these processes – every process spawned by an already marked process will again bear the same identifier. Using the device whitelist controller with the cgroup mechanism, ACLs on devices are implemented. Libvirtd uses cgroups to restrict access of each virtual machine process to only the devices assigned to this virtual machine, even though ordinary UNIX permission bits would have granted access to these devices. Please note that this mechanism only applies when the disk resource granted to a virtual machine is based on iSCSI, LVM or SANs. It does not apply to backends like regular files, NFS or others.

Virtual machines are associated with one or more unique IP addresses that can be used to communicate with other virtual machines on the same host or with other external entities. The TOE ensures that the configured IP addresses are used by the virtual machines for any network-related communication.

Management of the virtual machines is established via the libvirtd daemon. This daemon is accessible via SSH and is restricted to users belonging to the group "libvirt" with the evaluated configuration defined for the TOE. All aspects of virtual machine management, including creation of virtual machines, assigning resources, starting, stopping, and destroying of virtual machines are offered via libvirtd.

This security function for virtual machine maintenance covers the SFRs FDP_IFC.2(VIRT), FDP_IFF.1(VIRT), FIA_UID.2(VIRT).

This security function for virtual machine communication covers all SFRs of FDP_ETC.2(VIRT), FDP_ITC.2(VIRT), FPT_TDC.1(VIRT).

This security function for virtual machine communication covers all SFRs of FMT_MSA.1(VIRT-CACP), FMT_MSA.1(VIRT-CIFCP), FMT_MSA.3(VIRT-CACP), FMT_MSA.3(VIRT-CIFCP), FMT_MTD.1(VIRT-COMP).

7.1.7.1 Required hardware support

The Linux kernel uses various hardware mechanisms to provide the virtualization support and ensures proper separation of virtual machines. The following list enumerates the hardware support and indicates whether the underlying hardware must provide the respective functionality to achieve full separation.

Processor virtualization support
The Linux kernel uses the Intel VT-x and the AMD AMD-V processor support to allow untrusted software to execute in user mode and supervisor mode. The KVM mechanism of the Linux kernel is only operational if these support mechanisms are implemented. This evaluation applies only when the processor virtualization support is present and enabled. The kernel marks the enabled virtualization support in /proc/cpuinfo as the CPU flag vmx (Intel) and svm (AMD).

Processor virtualization support (IBM System Z)
The Linux kernel uses the IBM System Z SIE instruction to allow untrusted software to execute in user mode and supervisor mode. The KVM mechanism of the Linux kernel is only operational if these support mechanisms are implemented. This evaluation applies only when the processor virtualization support is present and enabled.

IOMMU virtualization support
If PCI device assignment is configured to be used, the hardware must provide the IOMMU mechanism located on the North Bridge chip set. Intel chip sets with the VT-d support as well as AMD chip sets with the HyperTransport architecture implement the IOMMU support. The KVM Linux kernel support can only implement the PCI device assignment functionality when the chip set implements the mentioned IOMMU mechanisms and have them enabled.

7.1.8 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 cases (like for /etc/shadow), the file is even readable to the root user only. It is the task of the persons responsible for setting up and administrating the system to ensure that

the access control features of the TOE are used throughout the lifetime of the system to protect those databases. These configuration files are accessed using applications which are able to interpret the contents of these configuration files. Each TOE instance maintains its own TSF database. Synchronizing those databases is not performed in the evaluated configuration. If such synchronization is required by an organization it is the responsibility of an administrative user of the TOE to achieve this either manually or with some automated assistance.

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

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

7.1.8.1 Privileges

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

Functionality accessible in user space are applications that can be invoked by users. 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.

8 Abbreviations, Terminology and References

8.1 Abbreviations

ACL
Access Control List

API
Application Programming Interface

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.

AppArmor
Linux kernel LSM module that is able to implement additional restrictions for executables.

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 Authoritative 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.

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 SPM

Object
: For Linux, objects are defined by SPM.

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 Ubuntu 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.

Subject
There are two classes of subjects in the TOE: 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 Ubuntu operating system, running and tested on the hardware and firmware specified in this Security Target. The BIOS firmware as well as the hardware are not part of 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 Ubuntu 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 3.1R5
Date April 2017
Location http://www.commoncriteriaportal.org/files/ccfiles/CCPART1V3.1R5.pdf
Location http://www.commoncriteriaportal.org/files/ccfiles/CCPART2V3.1R5.pdf
Location http://www.commoncriteriaportal.org/files/ccfiles/CCPART3V3.1R5.pdf

FIPS180-4 **Secure Hash Standard**
Date March 2012
Location http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf

FIPS186-5DRAFT **FIPS 186-5 (Draft) Digital Signature Standard**
Date October 2019
Location https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-5-draft.pdf

FIPS197 **Advanced Encryption Standard**
Date 2001-11-26
Location http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

RFC3526 **More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)**
Date May 2003
Location http://tools.ietf.org/html/rfc3526

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

RFC4419 **Diffie-Hellman Group Exchange for the Secure Shell (SSH) Transport Layer Protocol**
Date March 2006
Location http://tools.ietf.org/html/rfc4419

RFC5647 **AES Galois Counter Mode for the Secure Shell Transport Layer Protocol**
Author(s) K. Igoe, J. Solinas
Date 2009-08-01
Location http://www.ietf.org/rfc/rfc5647.txt

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

RFC6668 **SHA-2 Data Integrity Verification for the Secure Shell (SSH) Transport Layer Protocol**
Date July 2012
Location http://tools.ietf.org/html/rfc6668

SSH25519 **Ed25519 public key algorithm for the Secure Shell (SSH) protocol**
Date February 3, 2018
Location https://tools.ietf.org/id/draft-ietf-curdle-ssh-ed25519-02.txt