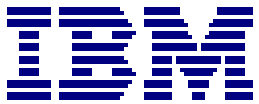

SuSE Linux Enterprise Server V 8 Security Target

Version: 1.6

Last Update: 2003-06-30

Table of Contents

References

[CC] Common Criteria for Information Technology Security Evaluation, CCIMB-99-031, Version 2.1, August 1999, Part 1 to 3

[CEM] Common Methodology for Information Technology Security Evaluation, CEM-99/045, Part 2 - Evaluation Methodology, Version 1.0, 1999

[GUIDE] ISO/IEC PDTR 15446 Title: Information technology – Security techniques – Guide for the production of protection profiles and security targets, ISO/IEC JTC 1/SC 27 N 2449, 2000-01-04

[CAPP] Controlled Access Protection Profile, Issue 1.d, 8 October 1999

[ITSEC] Information Technology Security Evaluation Criteria, Version 1.2, CEC, June 1991

[SECGUIDE] SLES Security Guide, distributed as part of the certification-sles-eal2 package

[ST-AIX-CC] AIX 5.2 Security Target

[TARGET] SuSE Linux Enterprise Server Security Target (this document)

1 Introduction

This is version 1.6 of the Security Target document for the evaluation of SuSE Linux Enterprise Server Version 8 with the certification-sles-eal2.rpm package.

1.1 ST Identification

Title: SuSE Linux Enterprise Server 8 Security Target, Version 1.6

Keywords: Linux, Open Source, general-purpose operating system, POSIX, UNIX.

This document is the security target for the CC evaluation of the SuSE Linux Enterprise Server (SLES 8) operating system product, and is conformant to the Common Criteria for Information Technology Security Evaluation [CC].

1.2 ST Overview

This security target documents the security characteristics of the SuSE Linux Enterprise Server operating system (Official name: SuSE Linux Enterprise Server Version 8) with the certification-sles-eal2.rpm package.

SuSE Linux Enterprise Server is a highly-configurable Linux-based operating system which has been developed to provide a good level of security as required in commercial environments. It also meets many of the requirements of the Controlled Access Protection Profile developed by the Information Systems Security Organization within the National Security Agency to map the TCSEC C2 class of the U.S. Department of Defence (DoD) Trusted Computer System Evaluation Criteria (TCSEC) to the Common Criteria framework. Full compliance with CAPP would require additional security functions as well as some modifications of existing security functions and therefore full compliance with the Controlled Access Protection Profile is not claimed.

Several servers running SuSE Linux Enterprise Server can be connected to form a networked system. The communication aspects within SuSE Linux Enterprise Server used for this connection are also part of the evaluation. It is assumed that the communication links themselves are protected against interception and manipulation by measures which are outside the scope of this evaluation.

This evaluation focuses on the use of the TOE as a server or a network of servers. Therefore a graphical user interface has not been included as part of the evaluation. In addition the evaluation assumes the operation of the network of servers in a non-hostile environment.

1.3 CC Conformance

This ST is CC *Part 2 conformant* and *Part 3 conformant*, with a claimed Evaluation Assurance Level of EAL2 augmented by ALC_FLR.1.

1.4 Strength of Function

The claimed strength of function for this TOE is: SOF-basic.

1.5 Structure

The structure of this document is as defined by [CC] Part 1 Annex C.

- Section 2 is the TOE Description.
- Section 3 provides the statement of TOE security environment.
- Section 4 provides the statement of security objectives.
- Section 5 provides the statement of IT security requirements.
- Section 6 provides the TOE summary specification, which includes the detailed specification of the IT Security Functions.
- Section 7 provides the Protection Profile claim

- Section 8 provides the rationale for the security objectives, security requirements and the TOE summary specification.

1.6 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.

SLES: This term serves as an abbreviation for "SuSE Linux Enterprise Server 8", which is the Target of this evaluation.

Administrative User: This term refers to an administrator of a SUSE Linux Enterprise Server system. Some administrative tasks requires use of the *root* username and password so that they can become the superuser (with a user ID of 0). Those users that have been asigned this capability are administrative users.

Authentication data: This includes the password for each user of the product. Authentication mechanisms using other authentication data than a password are not supported in the evaluated configuration.

Named Object: In SLES those objects that are subject to discretionary access control, which are file system objects and IPC objects.

Object: In SLES, objects belong to one of three categories: file system objects, IPC objects, and memory objects.

Product: The term product is used to define software components that comprise the SLES system.

Role: A role represents a set of actions that an authorized user, upon assuming the role, can perform. In this TOE only the roles of administrative user and normal user are supported.

Security Attributes: As defined by functional requirement FIA_ATD.1, the term ‘security attributes’ includes the following as a minimum: user identifier; group memberships; user authentication data.

Subject: There are two classes of subjects in SLES:

- untrusted internal subject - this is a SLES process running on behalf of some user, running outside of the TSF (for example, with no privileges).
- trusted internal subject - this is a SLES process running as part of the TSF. Examples are service daemons and the process implementing the identification and authentication of users.

System: Includes the hardware, software and firmware components of the SLES product which are connected/networked together and configured to form a usable system.

Target of Evaluation (TOE): The TOE is defined as the SuSE Linux Enterprise Server Version 8 operating system, running and tested on the hardware and firmware specified in this Security Target. The BootPROM firmware as well as the hardware form part of the TOE Environment.

User: Any individual/person who has a unique user identifier and who interacts with the SLES product.

2 TOE Description

The target of evaluation (TOE) is the operating system SuSE Linux Enterprise Server Version 8 (SLES) with the certification-sles-eal2.rpm package.

SLES is a general purpose, multi-user, multi-tasking Linux based operating system. It provides a platform for a variety of applications in the governmental and commercial environment. SLES is available on a broad range of computer systems, ranging from departmental servers to multi-processor enterprise servers.

SLES is based on United Linux, which is a common effort of several organizations to develop a common Linux platform designed as an enterprise platform for server applications.

The SLES evaluation covers a potentially distributed, but closed network of IBM xSeries servers running the evaluated version of SLES. 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 SLES that run in kernel mode plus some trusted processes. These are the functions that enforce the security policy as defined in this Security Target. Tools and commands executed in user mode that are used by an administrative user need also to be trusted to manage the system in a secure way. But as with other operating system evaluations they are not considered to be part of this TSF.

Also the hardware and the BootProm firmware is considered not to be part of the TOE but part of the TOE environment.

The TOE includes installation from CDROM and from a local hard disk partition.

The TOE includes standard networking applications, such as ftp and ssh. xinetd is used to protect network applications which might otherwise have security exposures.

System administration tools include the standard commands. 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 HTTP server using a port above 1024 (e. g. on port 8080) may be used as a normal application running without root privileges on top of the TOE.

2.1 Intended Method of Use

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 uses the standard Unix model of normal (unprivileged) users and administrative users that have the capability to get full root privileges. So, whenever this Security Target mentions the administrative user role it is identical to the term "root".

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

The TOE permits one or more processors and attached peripheral and storage devices to be used by multiple users to perform a variety of functions requiring controlled shared access to the data stored on the system. Such installations are typical for workgroup or enterprise computing systems accessed by users local to, or with otherwise protected access to, the computer system.

It is assumed that responsibility for the safeguarding of the data protected by the TOE can be delegated to the TOE users. All data is under the control of the TOE. The data is stored in named objects, and the TOE can associate with each controlled object a description of the access rights to that object.

All individual users are assigned a unique user identifier within the single host system that forms the TOE. This user identifier is used as the basis for access control decisions. The TOE authenticates the claimed identity of the user before allowing the user to perform any further actions.

The TOE 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 or administrative users. Ownership of named objects may be transferred under the

control of the access control policy.

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

SuSE Linux Enterprise Server has significant security extensions compared to standard UNIX systems:

- Access Control Lists,
- A Journaling File System,
- Integrated authentication framework (PAM). The following PAM modules are included in the evaluated configuration and implement security functions:
  - pam_unix2.so (basic password based authentication, configured to use MD5)
  - pam_pwcheck.so (use of cracklib to ensure strong passwords)
  - pam_wheel.so (to restrict the use of the su command to members of the trusted group)
  - pam_tally.so (to limit the number of consecutive unsuccessful authentication attempts)
  - pam_nologin.so (to check /etc/nologin)
  - pam_securetty.so (to restrict root access to specific terminals)

  In addition for some commands that require user authentication (e. g. chage) the module pam_rootok.so may be used to avoid that an administrative user with the effective user ID of root has to re-enter the password.

2.2 Summary of Security Features

The primary security features of the product are:

- Identification and Authentication
- Discretionary Access Control
- Object reuse functionality
- Security Management
- TSF Protection.

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

2.2.1 Identification and Authentication

SLES provides identification and authentication using pluggable authentication modules (PAM) based upon user passwords. The quality of the passwords used can be enforced through configuration options controlled by SLES. Other authentication methods (e. g. Kerberos authentication, token based authentication) that are supported by SLES as pluggable authentication modules are not part of the evaluated configuration. Functions to ensure a basic password strength and limit the use of the su command and restrict root login to specific terminals are also included.

2.2.2 Discretionary Access Control

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

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

2.2.3 Object Reuse

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

2.2.4 Security Management

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

2.2.5 TSF Protection

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

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

The TOE and the hardware and firmware components are required to be physically protected from unauthorized access. The system kernel mediates all access to the hardware mechanisms themselves, other than program visible CPU instruction functions.

2.3 Software

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

- SuSE Linux Enterprise Server Version 8 with the certification-sles-eal2.rpm package

The TOE and its documentation is supplied on CD-ROM except for the certification-sles-eal2.rpm package which needs to be downloaded from the SuSE web site. This package contains the Security Guide, all packages that have been updated to fix problems and scripts that can be used for the secure installation process. The user needs to verify the integrity and authenticity of those packages using the standard package verification procedure as described in the manuals distributed with the product.

The following list of packages that make up the TOE in the evaluated configuration. This includes packages that contribute to the TSF as well as packages that contain untrusted user programs from the distribution. Note that additional untrusted user programs may be installed and used as long as they are not setuid or setgid to root.

aaa_base	grub	openssl	util-linux
aaa_skel	gzip	pam	vim
acl	hdparm	pam-modules	vsftpd
ash	heimdal-lib	parted	w3m
at	howtoenh	pciutils	wget
attr	hwinfo	pcre	xinetd
bash	iproute2	perl	yast2
bc	iputils	permissions	yast2-bootloader
bzip2	isapnp	popts	yast2-core
certification-sles-eal2	k_deflt	postfix	yast2-country
cpio	k_smp	ps	yast2-installation
cracklib	kbd	readline	yast2-mouse
cron	ksymoops	rpm	yast2-ncurses
curl	l2h-pngicons	sed	yast2-network

cyrus-sasl	less	shadow	yast2-online-update
db	libgcc	sh-utils	yast2-packagemanager
devs	libstdc++	sitar	yast2-packager
dialog	libxcrypt	sles-admin-x86+x86-64_en	yast2-pam
diffutils	libxml2	sles-inst-x86+x86-64_en	yast2-runlevel
e2fsprogs	liby2util	sles-release	yast2-security
ed	logrotate	star	yast2-storage
file	lprng	suse-build-key	yast2-sysconfig
filesystem	lukemftp	sysconfig	yast2-theme-SuSELinux
fileutils	m4	syslogd	yast2-theme-UnitedLinux
fillup	mailx	sysvinit	yast2-trans-en_US
findutils	man	tar	yast2-transfer
freetype2	man-pages	telnet	yast2-update
gawk	mktemp	terminfo	yast2-users
gdbm	modutils	texinfo	yast2-xml
glibc	ncurses	textutils	zlib
gpg	netcat	timezone	
gpm	netcfg	UnitedLinux-build-key	
grep	openldap-client	unitedlinux-release	
groff	openssh	utempter	

2.4 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 Security Guide and installed accordingly.
- SLES supports the use of IPv4 and IPv6, only IPv4 is included.
- Both installation from CD and installation from a defined disk partition are supported.
- The default configuration for identification and authentication are the defined password based PAM modules. Support for other authentication options e.g. smartcard authentication, is not included in the evaluation configuration.
- If the system console is used, it must be connected directly to the workstation and afforded the same physical protection as the workstation.

The TOE comprises a single server machine (and optional peripherals) listed in section 2.4.2 running the system software listed the package list in section 2.3 (a server running the above listed software is referred to as a “TOE server” below).

Several TOE servers may be interlinked by a LANs, which may be joined by bridges/routers or by TOE workstations which act as routers/gateways. But one has to keep in mind that all servers within this network implement their own security policy. No synchronization function for those policies exists. As a result a single user may have user accounts on each of those servers which may have different user IDs, different roles and other attributes. If those are required to be syncronized for the different servers this syncronization has to be performed in the TOE environment.

If other systems are connected to the network they need to be configured and managed by the same authority using an appropriate security policy not conflicting with the security policy of the TOE.

2.4.1 File systems

The following file system types are supported:

- Ext3 journaling filesystem,
- the ISO 9660 filesystem for CD-ROM drives,
- The process file system, *procfs (/proc)* , provides access to the process image of each process on the machine as if the process were a “file”. Process access decisions are enforced by DAC attributes inferred from the underlying

process' DAC attributes.

2.4.2 Technical Environment for Use

The following assumptions about the technical environment the TOE is intended to be used in are made:

a) The TOE is running on the following hardware platforms:

- IBM xSeries Systems using Intel Pentium 4 or XEON processors

b) The following peripherals can be run with the TOE preserving the security functionality:

- all terminals and printers supported by the TOE (except hot pluggable devices connected via USB or IEEE 1394 (Firewire) interfaces)
- all storage devices and backup devices supported by the TOE (hard disks, CDROM drives,streamer drives, floppy disk drives) (except hot pluggable devices connected via USB or IEEE 1394 (Firewire) interfaces)
- all Ethernet and Token-Ring network adapters supported by the TOE

Pheripheral devices connected via PCMCIA are not supported.

3 TOE Security Environment

3.1 Introduction

The statement of TOE security environment describes the security aspects of the environment in which the TOE is intended to be used and the manner in which it is expected to be deployed.

To this end, the statement of TOE security environment identifies the list of assumptions made on the operational environment (including physical and procedural measures) and the intended method of use of the product, defines the threats that the product is designed to counter, and the organisational security policies with which the product is designed to comply.

3.2 Threats

The assumed security threats are listed below.

The **IT assets** to be protected comprise the information stored, processed or transmitted by the TOE. The term “information” is used here to refer to all data held within a server, including data in transit between workstations.

The TOE counters the general threat of unauthorized access to information, where "access" includes disclosure, modification and destruction.

The **threat agents** can be categorized as either:

- unauthorized users of the TOE, i.e. individuals who have not been granted the right to access the system; or
- authorized users of the TOE, i.e. individuals who have been granted the right to access the system.

The threat agents are assumed to originate from a well managed user community in a non-hostile working environment, and hence the product protects against threats of inadvertent or casual attempts to breach the system security. The TOE is not intended to be applicable to circumstances in which protection is required against determined attempts by hostile and well funded attackers with a medium or high level of expertise to breach system security.

The threats listed below are grouped according to whether or not they are countered by the TOE. Those that are not countered by the TOE are countered by environmental or external mechanisms.

3.2.1 Threats countered by the TOE

T.UAUSER An attacker (possibly, but not necessarily, an unauthorized user of the TOE) may impersonate an authorized user of the TOE. This includes the threat of an authorized user that tries to impersonate as another authorized user without knowing the authentication information.

T.UAACCESS An authorized user of the TOE may access information resources without having permission from the person who owns, or is responsible for, the information resource for the type of access.

3.2.2 Threats to be countered by measures within the TOE environment

The following threats to the system need to be countered in the TOE environment:

TE.HWMF A user (normal or administrative) is losing stored data due to hardware malfunction.

TE.COR_FILE Security enforcing or relevant files of the TOE are manipulated or accidentally corrupted without an administrative user being able to detect this.

TE.HW_SEP The underlying hardware functions of the hardware the TOE is running on does not provide sufficient capabilities to support the self-protection of the TSF from unauthorized programs.

3.3 Organizational Security Policies

The TOE complies with the following organizational security policies:

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

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

a) the owner of the object; and

b) the identity of the subject attempting the access; and

c) the implicit and explicit access rights to the object granted to the subject by the object owner or an administrative user.

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

3.4 Assumptions

This section indicates the minimum physical and procedural measures required to maintain security of the SLES 8 product.

3.4.1 Physical Aspects

A.LOCATE The processing resources of the TOE will be located within controlled access facilities which will prevent unauthorized physical access.

A.PROTECT The TOE hardware and software critical to security policy enforcement will be protected from unauthorized physical modification.

3.4.2 Personnel Aspects

A.MANAGE It is assumed that there are one or more competent individuals who are assigned to manage the TOE and the security of the information it contains.

A.NO_EVIL_ADMIN The system administrative personnel are not careless, willfully negligent, or hostile, and will follow and abide by the instructions provided by the administrator documentation.

A.COOP 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.UTRAIN Users are trained well enough to use the security functionality provided by the system appropriately.

A.UTRUST Users are trusted to accomplish some task or group of tasks within a secure IT environment by excercising complete control over their data.

3.4.3 Connectivity Aspects

A.NET_COMP All network components (like bridges and routers) are assumed to correctly pass data without modification.

A.PEER Any other systems with which the TOE communicates are assumed to be under the same management control and operate under the same security policy constraints. There are no security requirements which address the need to trust external systems or the communications links to such systems.

A.CONNECT All connections to peripheral devices and all network connections reside within the controlled access facilities. Internal communication paths to access points such as terminals or other systems are assumed to be adequately protected.

4 Security Objectives

4.1 Security Objectives for the TOE

O.AUTHORIZATION The TOE must ensure that only authorized users gain access to the TOE and its resources.

O.DISCRETIONARY_ACCESS The TSF must control access to resources based on identity of users. The TSF must allow authorized users to specify which resources may be accessed by which users.

O.RESIDUAL_INFO The TOE must ensure that any information contained in a protected resource is not released when the resource is recycled.

O.MANAGE The TSF must provide all the functions and facilities necessary to support admnistrative users that are responsible for the management of TOE security and must ensure that only administrative users are able to access such functionality.

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

4.2 Security Objectives for the TOE Environment

All security requirements listed in this section are targeted at the non-IT environment of the TOE.

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

OE.CREDEN Those responsible for the TOE must ensure that user authentication data is stored securely and not disclosed to unauthorized individuals. In particular:

Procedures must be established to ensure that user passwords generated by an administrator during user account creation or modification are distributed in a secure manner, as appropriate for the purpose of the system.

The media on which authentication data is stored must not be physically removable from the system by other than administrative users.

Users must not disclose their passwords to other individuals.

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.

OE.PHYSICAL Those responsible for the TOE must ensure that those parts of the TOE critical to security policy are protected from physical attack which might compromise IT security objectives.

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:

DAC protections on security critical files (such as configuration files and authentication databases) shall always be set up correctly.

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.

This requires that users are trained to perform those tasks properly and trustworthy to not deliberately misuse their access to information and pass it on to somebody that does not have the right to access the information.

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

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 (i.e., security) compromise is obtained.

OE.SOFTWARE_IN Those responsible for the TOE shall ensure that the system shall be configured so that only an administrative user can introduce new trusted software into the system.

OE.SERIAL_LOGIN Those responsible for the TOE shall implement procedures to ensure that users clear the screen before logging off where serial login devices (e.g.IBM 3151 terminals) are used.

OE.HW_SEP The underlying hardware must provide separation mechanism that can be used by the TOE to protect the TSF and TSF data from unauthorized access and modification.

The following security objective applies in environments where specific threats to networked systems need to be countered. (Either physical protection measures or cryptographic controls may be applied to achieve this objective, but they are **not** part of the security functions of TOE defined in this Security Target.)

OE.PROTECT Those responsible for the TOE must ensure that procedures and/or mechanisms exist to ensure that data transferred between servers is secured from disclosure, interruption or tampering.

5 Security Requirements

5.1 TOE Security Functional Requirements

5.1.1 User Data Protection (FDP)

5.1.1.1 Discretionary Access Control Policy (FDP_ACC.1)

FDP_ACC.1 The TSF shall enforce the **Discretionary Access Control Policy** on **processes acting on the behalf of users as subjects and file system objects (ordinary files, directories, symbolic links, device special files, UNIX Domain socket special files, named pipes), IPC objects (message queues, semaphores, shared memory segments) and all operations among subjects and objects covered by the DAC policy.**

5.1.1.2 Discretionary Access Control Functions (FDP_ACF.1)

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

a) **The effective user identity and group membership(s) associated with a subject; and**

a) **The following access control attributes associated with an object:**

File system objects:

POSIX ACLs and permission bits.
(ACLs can be used to grant or deny access to the granularity of a single user or group using Access Control Entries. Those ACL entries include the standard Unix permission bits. Posix ACLs can be used for file system objects within the ext3 file system).

Access rights for file system objects are:
- read
- write
- execute (ordinary files)
- search (directories)

IPC objects:

permission bits

Access rights for IPC objects are:
- read
- write

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

File system objects within the ext3 file system:

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

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

Or

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

Or

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

File system objects in other file systems:

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

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

Or

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

Or

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

IPC objects:

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

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

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

File System Objects:

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

IPC objects:

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

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

Write access to file system objects on a file system mounted as read-only is always denied.
Write access to a file marked as immutable is always denied.

5.1.1.3 Object Residual Information Protection (FDP_RIP.2)

FDP_RIP.2 The TSF shall ensure that any previous information content of a resource is made unavailable upon the **allocation of the resource to** all objects.

5.1.2 Identification and Authentication(FIA)

5.1.2.1 User Attribute Definition (FIA_ATD.1)

FIA_ATD.1 The TSF shall maintain the following list of security attributes belonging to individual users:

a) **User Identifier;**

b) **Group Memberships;**

c) **Authentication Data;**

d) **Security-relevant Roles; and**

e) **no other attributes**

5.1.2.2 Strength of Authentication Data (FIA_SOS.1)

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

a) **For each attempt to use the authentication mechanism, the probability that a random attempt will succeed is less than one in 1,000,000;**

b) **For multiple attempts to use the authentication mechanism during a one minute period, the probability that a random attempt during that minute will succeed is less than one in 100,000; and**

c) **Any feedback given during an attempt to use the authentication mechanism will not reduce the probability below the above metrics.**

5.1.2.3 Authentication (FIA_UAU.2)

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

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

5.1.2.4 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.

5.1.2.5 Identification (FIA_UID.2)

FIA_UID.2.1 The TSF shall require each user to identify itself before allowing any other TSF-mediated actions on behalf of that user.

5.1.2.6 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:

Application Note: The user attributes associated with the subject are:

a) The user identity or identities which are used to enforce the Discretionary Access Control Policy;

b) The group membership or memberships used to enforce the Discretionary Access Control Policy;

In addition the TSF shall enforce the following rules on the initial association of user security attributes with subjects acting on the behalf of a user:

a) Upon successful identification and authentication, the real user identifier and the effective user identifier shall be those specified in the user entry for the user that has authenticated successfully.

b) Upon successful identification and authentication, the real group identifier and the effective group identifier shall be those specified via the group membership attribute in the user entry.

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

a) The effective user ID of a user can be changed by the use of an executable with the setuid bit set. In this case the program is executed with the effective user ID of the program owner. Access rights are then evaluated using the effective user ID of the program owner.

b) The effective user ID of a user can be changed by the su command. In this case the effective user ID of the user is changed to the user specified in the su command (provided authentication is successful).

c) The effective group ID of a user can be changed by the use of an executable with the setgid bit set. In this case the program is executed with the effective group ID of the program owner. Access rights are then evaluated using the effective group ID of the program owner.

5.1.3 Security Management (FMT)

5.1.3.1 Management of Object Security Attributes (FMT_MSA.1)

FMT_MSA.1.1 The TSF shall enforce the **Discretionary Access Control Policy** to restrict the ability to **modify** the access control attributes **associated with a file system object or IPC object** to **administrative users and the owner of the object. For IPC objects also the original creator of the object has the ability to modify the access control attributes**

5.1.3.2 Static Attribute Initialization (FMT_MSA.3)

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

FMT_MSA.3.2 The TSF shall allow the **administrative users and the owner of the object** to specify alternative initial values to override the default values when an object or information is created.

Application Note: The term SFP in FMT_MSA.3.1 in Volume 2 of the Common Criteria is printed in italics but is not as one would expected stated as "[assignment: *SFP*]". It is assumed that such an assignment was intended by the authors of the CC and has therefore been performed here.

5.1.3.3 Management of User Attributes (FMT_MTD.1)

FMT_MTD.1.1 The TSF shall restrict the ability to **initialize and modify** the **user security attributes**, **other than authentication data,** to **administrative users**.

5.1.3.4 Management of Authentication Data (FMT_MTD.1)

FMT_MTD.1.1 The TSF shall restrict the ability to **initialize** the **authentication data** to **administrative users**.

FMT_MTD.1.1 The TSF shall restrict the ability to **modify** the **authentication data** to **the following:**

a) **administrative users; and**

b) **normal users, which are allowed to modify their own authentication data**

5.1.3.5 Revocation of User Attributes (FMT_REV.1)

FMT_REV.1.1 The TSF shall restrict the ability to revoke security attributes associated with the **users** within the TSC to **administrative users**.

FMT_REV.1.2 The TSF shall enforce the rules:

a) **Revocations/modifications made by an administrative user to security attributes of a user like the user identifier, user name, user group(s), user password or user login shell shall be effective the next time the user logs in**.

Application Note: Many security-relevant authorizations could have serious consequences if misused, so an immediate revocation method must exist, although it need not be the usual method (e.g., The usual method may be editing the trusted users profile, but the change doesn't take effect until the user logs off and logs back on. The method for immediate revocation might be to edit the trusted users profile and "force" the trusted user to log off.). The immediate method must be specified in the ST and in administrator guidance.

5.1.3.6 Revocation of Object Attributes (FMT_REV.1)

FMT_REV.1.1 The TSF shall restrict the ability to revoke security attributes associated with the **objects** within the TSC to **users authorized to modify the security attributes by the Discretionary Access Control policy**

FMT_REV.1.2 The TSF shall enforce the rules:

a) **Access rights to file system and IPC objects are checked when the object is**

opened. Revocations of access rights for file system objects become effective the next time a user affected by the revocation tries to open a file system object.

5.1.3.7 Specification of Management Functions (FMT_SMF.1)

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

- **Object security attributes management**
- **User attribute management**
- **Authentication data management**

Application Note: This security functional requirement has been added as a result of AIS 32, Final Interpretation 065. The security functional requirement was added because a dependency from FMT_MSA.1 and FMT_MTD.1 to this new component has been defined in AIS 32, Final Interpretation 065.

5.1.3.8 Security Management Roles (FMT_SMR.1)

FMT_SMR.1.1 The TSF shall maintain the roles:

a) **administrative users;**

b) **normal users**

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

Application Note: The role model supported by the TOE is a very simple one: the administrative user is root (extended to all members of the trusted group that may su to root). All other users of the system have the user role.

5.1.4 Protection of the TOE Security Functions (FPT)

5.1.4.1 Reference Mediation (FPT_RVM.1)

FPT_RVM.1.1 The TSF shall ensure that the TSP enforcement functions are invoked and succeed before each function within the TSC is allowed to proceed.

5.1.4.2 Domain Separation (FPT_SEP.1)

FPT_SEP.1.1 The TSF shall maintain a security domain for its own execution that protects it from interference and tampering by untrusted subjects.

FPT_SEP.1.2 The TSF shall enforce separation between the security domains of subjects in the TSC.

5.1.4.3 Strength of Function

The claimed minimum strength of function is *SOF-basic*.

Note: The only security function within the TOE that uses a statiscal or probabilistic mechanism is the authentication function that uses passwords.

5.2 TOE Security Assurance Requirements

The target evaluation assurance level for the product is EAL2 [CC] augmented by ALC_FLR.1.

5.3 Security Requirements for the IT Environment

The only IT environment where requirements are stated is the underlying processor, that has to provide the mechanism

to protect the TSF and TSF data from unauthorized access and tampering. This is expressed with the following security functional requirement for the processor used to execute TOE software:

FDP_ACC.1 Subset access control

FDP_ACC.1.1 The TSF shall enforce the **memory access control policy** on **instructions as subjects and memory locations and processor register as objects.**

FDP_ACF.1 Security attribute based access control

FDP_ACF.1.1 The TSF shall enforce the **memory access control policy** to objects based on **the processor state (user or supervisor).**

FDP_ACF.1.2 The TSF 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**.

Application Note: The precise definition of the objects and the rules for the access control policy differ slightly depending on the processor type. For this security requirement on the IT environment the definition is detailed enough, since the implementation is not checked in this evaluation. When used for the hardware evaluation of a real processor those rules have to be stated precisely.

FDP_ACF.1.3 The TSF shall explicitly authorise 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 user mode.**

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

FMT_MSA.3 Static attribute initialisation

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

FMT_MSA.3.2 The TSF 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 start-up. They have to be „permissive", since the initialization routine needs to set up the memory management unit and the device register etc.. With respect to the hardware there is no „role" model implemented but the access control policy is purely based on a single attribute („user" or „supervisor" state) that can not be managed or assigned to a „user". The attribute changes under well defined conditions (when the processor encounters an exception, an interrupt or when the a call gate for a higher ring of privilge is called. The security requirement FMT_MSA.1 was therefore not applicable because the security attribute can

not 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.

Note: OE.PROTECT mentions cryptographic controls as one possible security function to meet this objective. But it also mentioned there that this objective can be fully met by physical protection features, which are then part of the non-IT environment. Therefore it is not mandatory to address this security objective by a security function in the IT environment.

5.4 Security Requirements for the Non-IT Environment

All the security objectives for the TOE environment address physical protection of the TOE or procedures that need to be obeyed by administrative users.

6 TOE Summary Specification

6.1 Security Enforcing Components Overview

6.1.1 Introduction

This chapter describes the security functions of SuSE Linux Enterprise Server Version 8 that are subject to this evaluation. Only a subset of the overall security related functions of SLES 8 has been included in this evaluation. Those functions provide some basic security for a single server within a protected environment. They allow for identification and authentication of users, access control to files and IPC objects and secure management. As such those functions are required as a basis for other security related functions and mechanisms that are implemented in SLES but not addressed in this evaluation.

As the first evaluation of a Linux based system, it was decided to limit the scope of the evaluation and focus on the challenge to get an Open Source based system through a formal evaluation process. Other security functions that will significantly enhance the usability of the evaluated configuration may then well be added in further evaluation efforts.

6.1.2 Kernel Services

The SLES kernel includes the base kernel and some kernel modules. The base kernel includes support for system initialization, memory management, file and I/O management, process control, and Inter-Process Communications (IPC) services. Kernel modules are dynamically loadable modules that the kernel will load on demand and that execute with kernel privileges.

Device drivers may be implemented as kernel modules.

The SLES kernel implements a virtual memory manager (VMM) that allocates a large, contiguous address space to each process running on the system. This address space is spread across physical memory and paging space on a secondary storage device.

The process management component includes the software that is responsible for creating, scheduling, and terminating processes and process threads. Process management allows multiple processes to exist simultaneously on a computer and to share usage of the computer's processor(s). A process is defined as a program in execution, that is, it consists of the program and the execution state of the program.

Process management also provides services such as inter-process communications (IPC) and event notification. The base kernel implements

- named pipes
- unnamed pipes
- signals
- semaphores
- shared memory
- message queues
- Internet domain sockets
- UNIX domain sockets

The file and I/O software provides access to files and devices. The SLES Virtual File System (VFS) provides a consistent view of multiple physical file system implementations. There are three different types of file systems included in the evaluated configuration: the journalled file system ext3, CDROM File System ISO-9660 (read-only), and the proc file system. ext3 and ISO-9660 work off of a physical medium (disk, CDROM). The proc file system does not represent or provide a physical data storage file system but is used as a configuration and monitoring interface to the kernel, provided by the kernel only in a running system. procfs also represents the abstraction of processes (tasks) being files. Processes / tasks are listed as files and directories containing live status information for each process in the system. Process access decisions are enforced by DAC attributes inferred from the underlying process' DAC attributes.

6.1.3 Non-Kernel TSF Services

The non-kernel TSF services are:

- Identification and Authentication services
- Network application layer services
- Configuration and management commands requiring root privileges

Those services support the security functions implemented within the kernel and use the kernel interface for this purpose, but they are not running themselves in kernel mode. Those functions are included in the TSF as far as they are required for the security services of the TOE (Identification and Authentication services), while other services that are implemented as tools or commands for the use of the administrative user and where the kernel prohibits the use misuse of those tools or commands since they use kernel functions restricted to administrative users and attempted use by normal users is prohibited by the kernel.

6.1.4 Network Services

The TOE is capable of providing the following types of services:

- Local services to the user currently logged in to the local computer console.
- Local services to previous users via deferred jobs.
- Local services to users who have accessed the local host via the network using protocols such as ftp or ssh.
- Network services to clients on either the local host or on remote hosts.

Network services are provided to clients via a client-server architecture. This client-server architecture refers to the division of the software that provides a service into a client portion, which makes requests, and a server portion, which carries out client requests (usually on a different computer). A service protocol acts as the interface between the client and server.

The primary low-level protocols are Internet Protocol (IP), Transmission Control Protocol (TCP), and User Datagram Protocol (UDP). IP is not user visible, but non-TSF processes may communicate with other hosts in a networked system using a reliable byte stream or unreliable datagrams, TCP and UDP respectively.

The higher-level network services are built on TCP or UDP. The application protocols provided using TCP and supporting user authentication and running on privileged ports are:

- Secure shell and file transfer services (SSH and FTP) are supported within the evaluated product.

Note that ssh is supported as part of the security functions of the TOE but the cryptographic functions implemented in this protocol (regardless if they are used for authentication, integrity protection or confidentiality protection) have not been analyzed for the simple reason to keep the scope of this evaluation small enough. As a consequence no claims on the strength and effectiveness of those functions are made in this Security Target.

6.1.5 Security Policy Overview

The TOE is a single SLES system running on one machine. Several of those systems may be interconnected via a local area network and exchange information using the network services. But one should keep in mind that the following statements hold:

- There is a Linux (SLES) kernel running on each host computer in the system.
- Identification and authentication (I&A) is performed locally by each host computer. Each user is required to LOGIN with a valid password and user identifier combination at the local workstation and also at any remote computer where the user can enter commands to a shell program (using ssh). User ID and password for one human user may be different on different hosts. User ID and password on one host system are not known to other host systems on the network and therefore a user ID is relevant only for the host where it it defined.
- Discretionary access control (DAC) is performed locally by each of the host computers and is based on user identity and group membership on this host. Each process has an identity (the user on whose behalf it is operating) and

belongs to one or more groups. All named objects have an owning user, an owning group and a DAC attribute, which is a set of permission bits. In addition, file system objects optionally have extended permissions also known as an Access Control List (ACL). The ACL mechanism is a significant enhancement beyond traditional UNIX systems, and permits control of access based on lists of users and/or groups to whom specific permissions may be individually granted or denied.

- Object reuse is performed locally, without respect to other hosts.
- Interrupt handling is performed locally, without respect to other hosts.
- Privilege is based on the root identity. All privileged processes (setuid root programs and programs run under the root identity) start as processes with all privileges enabled. Unprivileged processes, which include setgid trusted processes, start and end with no privileges enabled.

6.1.6 TSF Structure

The TSF is the portion of the system that is responsible for enforcing the system's security policy. The TSF of SLES consists of two major components: kernel software and trusted processes. All these components must operate correctly for the system to be trusted. Those functions are supported by the mechanisms of the underlying hardware which are used to protect the TSF from tampering by untrusted processes.

The SLES hardware components support two execution states where kernel mode or supervisor state, software runs with hardware privilege and user mode or problem state software runs without hardware privilege. SLES also provides two types of memory protection: segmentation and page protection. The memory protection features isolate critical parts of the kernel from user processes and ensure that segments in use by one process are not available to other processes. The two-state architecture and the memory protections form the basis of the argument for process isolation and protection of the TSF.

The trusted processes include programs such as Linux administrative programs, scripts, shells, and standard Linux utilities that run with administrative privilege, as a consequence of being invoked by a user with administrative privileges. Non-kernel TSF software also includes daemons that provide system services, such as networking, as well as setuid and setgid programs that can be executed by untrusted users.

6.1.7 TSF Interfaces

Each subsection here summarizes a class of interfaces in the SLES operating system, and characterizes them in terms of the TSF boundary. The TSF boundary includes some interfaces, such as commands implemented by privileged processes, which are similar in style to other interfaces that are not part of the TSF boundary and thus not trusted. Some interfaces are part of the TSF boundary only when used in a privileged environment, such as an administrative user's process, but not when used in a non-privileged environment, such as a normal user process. All interface classes are described in further detail in the next chapter, and the mechanisms in subsequent chapters. As this is only an introduction, no explicit forward references are provided.

6.1.7.1 User Interfaces

The typical interface presented to a user is the command interpreter, or shell. The user types commands to the interpreter, and in turn, the interpreter invokes programs. The programs execute hardware instructions and invoke the kernel to perform services, such as file access or I/O to the user's terminal. A program may also invoke other programs, or request services using an IPC mechanism. Before using the command interpreter, a user must log in.

The command interpreter or shell as well as other programs operating on behalf of a user have the following interfaces:

- CPU instructions, which a process uses to perform computations within the processor's registers and a process's memory areas. CPU instructions are interpreted by the hardware, which is part of the TOE environment; CPU instructions are therefore not a TSF interface.
- System calls (e.g. open, fork), through which a process requests services from the kernel, which are invoked using a special CPU instruction. System calls are the primary way for a program operating on behalf of a user to request services of the TOE including the security services. System calls related to security functions are therefore part of the TSF interface.
- Directly-invoked trusted processes (e.g. passwd) which perform higher-level services, and are invoked with an exec

system call that names an appropriate program which is part of the TSF, and replaces the current process's content with it; a limited number of those processes exist that perform security functions and are therefore part of the TSF interface.

- Daemons, which accept requests stored in files or communicated via other IPC mechanisms, generally created through use of directly invoked processes (some trusted, some untrusted). A few daemons peform security functions and therefore part of the TSF interface.
- Network Services, (ssh, ftp). The network services interface operates at many different levels of abstraction. At the highest level, it provides a means for users on one host to request a virtual terminal connection on another host within the system. At a lower level, it allows a host on a networked system to request a specific service from another host within the system on behalf of a user. Examples of requested services include remotely login into the TOE and obtaining a shell or transfering whole files. At the lowest level, it allows a subject on one host in the system to request a connection (i.e. TCP), or deliver data (i.e. UDP) to a listening subject. Network services usually consist of a client on the requestor's side and a server (usually a daemon) running on the server's side. Authentication (if required by the service) and access control use dedicated interfaces to the functions on the server side which are therefore part of the TSF interface. Note that for the TOE only ssh and ftp are seen as TSF, because they use privileged ports and require user identification and authentication.

  Note: Users may start programs using unprivileged ports, but those programs operate with the effective userid of the calling user and are therefore restricted by the security policy of the TOE. Those user programs using unprivileged ports are not part of the TSF.

6.1.7.2 Operation and Administrator Interface

The primary administrative interfaces to SLES are the same as the interfaces for ordinary users; the administrative user logs into the system with a standard, untrusted, identity and password, and after assuming the root identity uses standard Linux commands to perform administrative tasks. Direct root login is only allowed from the system console (to avoid a denial of service attack).

The part of the administrative database (which is the set of all security relevant configuration files) that is used to configure and manage TSF is seen as part of the TSF interface. The administrative database is protected by the access control mechanisms of the TOE. It is therefore very important to set the access rights to the files of the administrative database such that non-administrative users are prohibited from modifying those files and have read access on a need to know basis only. Note that each server in the system has its own administrative database and if syncronization between those TSF database is required by the organization's security policy, it has to be done manually in the system environment. The TOE does not provide any function to syncronize TSF databases on different systems.

6.1.8 Secure and Non-Secure States

The secure state for the SLES is defined as a host's entry into multi-user mode with the administrative databases configured with the required access rights. At this point, the host accepts user logins and services network requests across the networked system. If these facilities are not available, the host is considered to be in a non-secure state. Although it may be operational in a limited sense and available for an administrative user to perform system repair, maintenance, and diagnostic activity, the TSF are not in full operation and is not necessarily protecting all system resources according to the security policy.

6.2 Description of the Security Enforcing Functions

6.2.1 Introduction

This chapter describes how the Security Enforcing components of the TOE provide the Security Requirements identified in chapter 5.

A high level description is provided for each group of security enforcing functions (SEF) providing a common feature or service, and stating how the functionality specified by the security enforcing function group is provided by the security enforcing components identified in this Chapter.

The security enforcing function groups identified in this chapter follow the description given in chapter 2:

- Identification and Authentication
- Discretionary Access Control
- Object Reuse
- Security Management
- TOE Protection

The TOE security functions (TSF) are described with sufficient detail to provide a general understanding of those functions and how they work. A more detailed description of those functions and a mapping of the TSF to TOE subsystems is provided in the high level design documentation.

References to components given in *italics* can be traced to manual pages or TOE sources for further information. Note also that some commands initiate trusted processes or are a local front end to a trusted process (e.g. *ftp* and the *ftpd* daemon, *ssh* and the *sshd* daemon). In these instances, a generic reference to the command is made.

6.2.2 Identification and Authentication (IA)

User identification and authentication in the SLES includes all forms of interactive login (e.g., using the *ssh* or *ftp* protocols) as well as identity changes through the *su* command. These all rely on explicit authentication information provided interactively by a user.

Identification and authentication of users is performed from a terminal where no user is logged on or when a user that is logged on starts a service that requires additional authentication. All those services use a common mechanism for authentication described in this chapter. They all use the administrative database. The administrative database is managed by administrative users, but normal users are allowed to modify their own password using the *passwd* command. This chapter also describes the authentication process for those network services that require authentication.

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. This section provides also a brief description how PAM is used and configured in the evaluated configuration.

The evaluated configuration supports password based login only (pam_unix2.so module). To strengthen the password used the pam_pwcheck.so module is deployed. To restrict the use of the su command to members of the trusted group the pam_wheel.so module is used.

The module pam_rootok.so allows a user with an effective userid of 0 to use several administrative commands without re-authentication.

The module pam_tally.so counts the number of consecutive unsuccessful authentication attempts for a user and blocks further login attempts for this user until an administrative user unblocks the user.

The module pam_securetty.so is used to restrict the login of root to a terminal listed in */etc/securetty*.

The module pam_nologin.so is used to allow to restrict login to root only (for example when critical system management activities need to be performed). If the file "nologin" exists in the /etc directory, the TOE rejects login attempts from any user except root and displays the message found in the file */etc/nologin* to users that try to log into the TOE.

6.2.2.1 User Identification and Authentication Data Management (IA.1)

Each server 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 syncronized on different servers. As a result the same user may have different usernames, different user IDs, different passwords and different attributes on different machines within the networked environment. Existing mechanism for syncronizing this within the whole networked system are not subject to this evaluation.

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

Users are allowed to change their passwords by using the *passwd* command, which is a setuid program with the owning userid of 0. This configuration allows a process running the *passwd* program to read the contents of */etc/shadow* and to

modify the */etc/shadow* file for the user's password entry, which would ordinarily be inaccessible to a non-privileged user process (IA1.1). Users are also forced to change their passwords at login time, if the password has expired (IA1.2).

The file */etc/passwd* conatins the user's name, the id of the user, an indicator, if the password of the user is valid, the principal group id of the user and a few other, not security relevant information (IA1.3). The encrypted password of the user itself is not stored in this file but in the file */etc/shadow* which can be protected against read access for ordinary users. This prohibits dictionary attacks on passwords in the passwd file as for example described in the paper of Ken Thomson and Bob Morris „Password Security - A Case History".

The file */etc/shadow* contains the MD5 encrypted password, the userid, the time the password was last changed and some other information that are not subject to the security functions as defined in this Security Target (IA1.4).

For a complete list of user attributes see the description of the function SM.

An administrative user can define the following restrictions on the login process (defined in /etc/login.defs):

- Delay in seconds before being allowed another attempt after a login failure
- Enable logging and display of /var/log/faillog login failure info
- Enable logging and display of /var/log/lastlog login time info.
- Maximum number of days a password may be used.
- Minimum number of days allowed between password changes.
- Minimum acceptable password length.
- Number of days a warning is given before a password expires.
- If compiled with cracklib support (as required by the evaluated configuration), where are the dictionaries
- Max number of login retries if password is bad
- Max time in seconds for login
- Maximum number of attempts to change password if rejected (password not compliant with the password policy)
- Number of significant characters in the password for crypt() (since MD5 is used, this is always 128 in the evaluated configuration)

This allows the administrative user to define restrictions on authentication data like the delay before another authentication attempt can be done, the minimum length of the password, checking the password against entries in a dictionary as well as the maximum life time of a password, the number of unsuccessful login attempts allowed before the account is locked (IA1.5). Those restrictions are stored in the file */etc/login.defs*. The administrative user can use those parameters to define a password policy such that the passwords satisfy the requirements defined in FIA_SOS.1.

Failed login attempts are recorded */var/log/faillog* (IA1.6) and the time of the last successful logins is recorded in */var/log/lastlog* (IA1.7).

In the evaluated configuration the above mentioned parameter need to be set in accordance with the following restrictions:

- Delay in seconds: greater or equal to 3 seconds
- Logging and display of /var/log/faillog and /var/log/lastlog info: enabled
- Maximum lifetime of a password: less than or equal to 60 days
- Minimum lifetime of a password: 1 day
- Minimum length of a password: 8 character
- Number of days a warning is given before password expires: 5 days
- Passwords found within the dictionaries for cracklib are not allowed
- Number of consecutive unsuccessful login retries: 3
- Maximum time for login: 60 seconds
- Maximum number of attempts to change the password: 3
- Number of significant characters in the password for the crypt function: any value between 16 and 128 (IA1.8)

This function contributes to satisfy the security requirements FIA_ATD.1, FIA_SOS.1, FMT_MTD.1 „User Attributes" and FMT_SMF.1.

6.2.2.2 Common Authentication Mechanism (IA.2)

SLES includes a common authentication mechanism which is a subroutine used for all activities that create a user session, including all the interactive login activities, batch jobs, and authentication for the SU command (IA2.1).

The common mechanism includes the following checks and operations:

- Check password authentication
- Check password expiration
- Check whether access should be denied due to too many consecutive authentication failures
- Get user security characteristics (e.g., user and groups)

The common I&A mechanism identifies the user based on the supplied user name, gets that user's security attributes, and performs authentication against the user's password.

This function contributes to satisfy the security requirements FIA_UAU.2 and FIA_UID.2.

6.2.2.3 Interactive Login and Related Mechanisms (IA.3)

The ssh and ftp as well as the su command used to change the effective user ID of a user all use the same authentication mechanism in the evaluated configuration (IA3.1). It is of course up to the remote system to protect the user's entry of a password correctly (e. g. provide only obscured feedback). As long as the remote system is also an evaluated version of the TOE, this is ensured by the security function of the TOE.

This function contributes to satisfy the security requirements FIA_UAU.2, FIA_UID.2 and FIA_UAU.7.

6.2.2.4 User Identity Changing (IA.4)

Users can change their identity (i.e., switch to another identity) using the *su* command (IA4.1). When switching identities, the real and effective user ID and real and effective group ID are changed to the one of the user specified in the command (after successful authentication as this user) (IA4.2). The primary use of the *su* command within the SLES 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 (IA4.3). In addition the use of the su command to switch to root has been restricted to users belonging to the trusted group (IA4.4). Users that don't have access to a terminal where root login is allowed and are not member of the trusted group will not be able to switch their real 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 is changed to that of the owner of the file containing the program while the real user ID remains that of the caller (IA4.5).

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

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

6.2.2.5 Login Processing (IA.5)

At the login process the real and effective user ID are set to the ID of the user that has logged in. Also with the su command the real and effective user ID and real and effective group ID are changed.

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

6.2.3 Discretionary Access Control (DA)

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

Details of the specific DAC policy applied to each type of resource are covered in the section "Discretionary Access

Control: File System Objects" and the section "Discretionary Access Control: IPC Objects".

Note: Signals are not subject to discretionary access control as described in this section of the Security Target. The rules when a process is allowed to send a signal to another process are not seen as security relevant and therefore not listed in this Security Target.

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

Finally, a subject with an effective user ID of 0 is exempt from all restrictions and can perform any action desired (DA0.1).

DAC provides the mechanism that allows users to specify and control access to objects that they own (DA0.2). DAC attributes are assigned to objects at creation time and remain in effect until the object is destroyed or the object attributes are changed (DA0.3). DAC attributes exist for, and are particular to, each type of object on SLES. DAC is implemented with permission bits and, when specified, ACLs.

A subject whose effective user ID matches the file owner ID can change the file attributes, the base permissions, and the extended permissions (except for read-only file systems, of course) (DA0.4). Changes to the file group are restricted to the owner and root (DA0.5).

The new file group identifier must either be the current effective group identifier or one of the group identifiers in the concurrent group set (DA0.6). In addition, a subject whose effective user ID is 0 can make any desired changes to the file attributes, the base permissions, the extended permissions, and owning user of the file (see DA0.1).

Permission bits are the standard UNIX DAC mechanism and are used on all SLES file system named objects (DA0.7). Individual bits are used to indicate permission for read, write, and execute access for the object's owner, the object's group, and all other users (i.e. world). The extended permission mechanism is supported only for file system objects within an ext3 file system and provides a finer level of granularity than do permission bits.

Write access is in general not granted for files on a file system mounted as read-only. Write access is also denied for files that have the immutable attribute.

6.2.3.1 Permission Bits (DA.1)

SLES supports standard UNIX permission bits to provide one form of DAC for file system objects in the /proc and ISO9660 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.

Each subject's access to an object is defined by some combination of these bits:

- rwx symbolizing read/write/execute
- r-x symbolizing read/execute
- r-- symbolizing read
- --- symbolizing null
  (DA1.1)

When a process attempts to reference an object protected only by permission bits, the access is determined as follows:

- Users with an effective user ID of 0 are able to read and write all files, ignoring the permission bits. Users with an effective user ID of zero are also able to execute any file if it is executable for someone.
- If the effective user ID = object's owning user ID and the owning user permission bits allow the type of access requested access is granted or denied with no further checks.
- If the effective group ID, or any supplementary groups of the process = object's owning group ID, and the owning group permission bits allow the type of access requested access is granted or denied with no further checks.
- If the process is neither the owner nor a member of an appropriate group and the permission bits for world allow the

type of access requested, then the subject is permitted access.

- If none of the conditions above are satisfied, and the process is not the root identity, then the access attempt is denied.
  (DA1.2)

This function contributes to satisfy the security requirements FDP_ACC.1 and FDP_ACF.1.

6.2.3.2 Access Control Lists supported by SLES (DA.2)

SLES provides support for POSIX type ACLs for the ext3 file system allowing to define a fine grained access control on a user basis. The semantics of those ACLs is summarized in this section.

An ACL entry contains the following information:

1. A tag type that specifies tha type of the ACL entry
2. A qualifier that specifies an instance of an ACL entry type
3. A permission set that specifies the discretionary access rights for processes identified by the tag type and qualifier
   (DA2.1)

6.2.3.2.1 ACL Tag Types

The following tag types exists:

1. ACL_GROUP
   an ACL entry of this type defines access rights for processes whose effective group ID or any supplementary group IDs match the one in the ACL entry qualifier
2. ACL_GROUP_OBJ
   an ACL entry of this type defines access rights for processes whose effective group ID or any supplementary group IDs match the group ID of the group of the file
3. ACL_MASK
   an ACL entry of this type defines the maximum discretionary access rights a process in the file group class
4. ACL_OTHER
   an ACL entry of this type defines access rights for processes whose whose attributes do not match any other entry in the ACL
5. ACL_USER
   an ACL entry of this type defines access rights for processes whose effective user ID matches the ACL entry qualifier
6. ACL_USER_OBJ
   an ACL entry of this type defines access rights for processes whose effective user ID matches the user ID of the owner of the file
   (DA2.2)

6.2.3.2.2 ACL Qualifier

The qualifier is required for ACL entries of type ACL_GROUP and ACL_USER and contain either the user ID or the group ID for which the access rights defined in the entry shall apply (DA2.3).

6.2.3.2.3 ACL Permissions

The permission that can be defined in an ACL entry are: read, write and execute/search (DA2.4).

6.2.3.2.4 Relation with File Permission Bits

An ACL contains exactly one entry for each of the ACL_USER_OBJ, ACL_GROUP_OBJ, and ACL_OTHER tag type (called the „required ACL entries”) (DA2.5). An ACL may have between zero and a defined maximum number of

entries of the type ACL_GROUP and ACL_USER (DA2.6).

An ACL that has only the three required ACL entries is called a „minimum ACL". ACLs with one or more ACL entries of type ACL_GROUP or ACL_USER are called an „extended ACL".

The standard UNIX file permission bits as described in the previous section are represented by the entries in the minimum ACL. The owner permission bits are represented by the entry of type ACL_USER_OBJ, the entry of type ACL_GROUP_OBJ represent the permission bits of the file's group and the entry of type ACL_OTHER represents the permission bits of processes running with an effective user ID and effective group ID or supplementary group ID different from those defined in ACL_USER_OBJ and ACL_GROUP_OBJ entries (DA2.7).

6.2.3.2.5 ACL_MASK

If an ACL contains an ACL_GROUP or ACL_USER type entry, then exactly one entry of type ACL_MASK is required in the ACL. Otherwise the entry of type ACL_MASK is optional (DA2.8).

6.2.3.2.6 Default ACLs

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 (DA2.9).

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.

6.2.3.2.7 Access Check Evaluation Algorithm

When a process attempts to reference an object protected by an ACL, it does so through a system call (e.g., open, exec). If the object has been assigned an ACL access is determined as according to the algorithm below:

ACCESS CHECK ALGORITHM
A process may request read, write, or execute/search access to a file system object protected by an ACL. The access check algorithm determines whether access to the object will be granted.

1. Write access to a file on a read-only file system will always be denied.

2. Write access to a file with the immutable attribute will always be denied.

3. **If** the effective user ID of the process matches the user ID of the file object owner, **then**

   if the ACL_USER_OBJ entry contains the requested permissions, access is granted,

   else access is denied.

4. **else if** the effective user ID of the process matches the qualifier of any entry of type ACL_USER, **then**

   if the matching ACL_USER entry and the ACL_MASK entry contain the requested permissions, access is granted,

   else access is denied.

5. **else if** the effective group ID or any of the supplementary group IDs of the process match the qualifier of the entry of type ACL_GROUP_OBJ, or the qualifier of any entry of type ACL_GROUP, **then**

   if the ACL_MASK entry and any of the matching ACL_GROUP_OBJ or ACL_GROUP entries contain all the requested permissions, access is granted,

else access is denied.

6. **else if** the ACL_OTHER entry contains the requested permissions,
   access is granted.

7. **else** access is denied.

(DA2.10)

This function contributes to satisfy the security requirement FDP_ACC.1 and FDP_ACF.1

6.2.3.2.8 DAC Revocation on 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 (DA2.11).

In cases where an administrative user determines that immediate revocation of access to a file system object is required, the administrative user can reboot the computer, resulting in a close on the object and forcing an open of the object on system reboot.

6.2.3.2.9 DAC: Directory

The execute permission bit for directories governs the ability to name the directory as part of a pathname. A process must have search (execute) access in order to traverse the directory during pathname resolution (DA2.12).

Directories may not be written directly, but only by creating, renaming, and removing (unlinking) objects within them. These operations are considered writes for the purpose of the DAC policy (DA2.13).

6.2.3.2.10 DAC: UNIX Domain Socket Special File

UNIX domain socket files are treated as files in the SLES file system from the perspective of access control, with the exception that using the bind or connect system calls requires that the calling process must have write access to the socket file (DA2.14).

UNIX domain sockets exist in the file system name space, the socket files can have both base mode bits and extended ACL entries (DA2.15).

UNIX domain sockets consist of a socket special file (managed by the File System) and a corresponding socket structure (managed by IPC). The TOE controls access to the socket based upon the caller's rights to the socket special file (DA2.16).

6.2.3.2.11 DAC: Named Pipes

Named pipes are treated identically to any other file in the SLES file system from the perspective of access control. Therefore permission bits and extended permissions can be used (DA2.17). For this reason named pipes are listed as file system objects (although they are used for interprocess communication). Note that named pipes follow the rules for IPC objects, if no ACLs are used (which probably is the normal case they are used).

6.2.3.2.12 DAC: Device Special File

The access control scheme described for file system objects is used for protection of character and block device special files (DA2.18). Most device special files are configured to allow read and write access by the root user, and read access by privileged groups. With the exception of terminal and pseudo-terminal devices and a few special cases (e.g., /dev/null and /dev/tty), devices are configured to be not accessible to normal users (DA2.19). The access mode of device files for ttys is changed during login time to read/write access of the user logging into the system; on logout the access rights are reset to allow only access by root (DA2.20).

This function contribute to satisfy the security requirement FDP_ACC.1, FDP_ACF.1, FMT_MSA.1, FMT_SMF.1, FMT_MSA.3 and FPT_SEP.1.

6.2.3.3 Discretionary Access Control: IPC Objects (DA.3)

6.2.3.3.1 DAC: Shared Memory

For shared memory segment objects (henceforth SMSs), access checks are performed when the SMS is initially attached, and are not checked on each subsequent access. Changes to access controls (i.e., revocation) are effective with the next attempt to attach to the SMS (DA3.1).

In cases where an administrative user determines that immediate revocation of access to a SMS is required, the administrative user can reboot the computer, thus destroying the SMS and all access to it.

If a process requests deletion of a SMS, it is not deleted until the last process that is attached to the SMS detaches itself (or equivalently, the last process attached to the SMS terminates) (DA3.2).

The default access control on newly created SMSs is determined by the effective user ID and group ID of the process that created the SMS and the specific permissions requested by the process creating the SMS (DA3.3).

- The owning user and creating user of a newly created SMS will be the effective user ID of the creating process (DA3.4).
- The owning group and creating group of a newly created SMS will be the effective group ID of the creating process (DA3.5).
- The creating process must specify the initial access permissions on the SMS, or they are set to null and the object is inaccessible until the owner sets them (DA3.6).
- SMSs do not have ACLs as described above, they only have permission bits (DA3.7).

Access permissions can be changed by any process with an effective user ID equal to the owning user ID or creating user ID of the SMS (DA3.8). Access permissions can also be changed by any process with an effective user ID of 0, also known as running with the root identity (DA3.9).

6.2.3.3.2 DAC: Message Queues

For message queues, access checks are performed for each access request (e.g., to send or receive a message in the queue) (DA3.10). Changes to access controls (i.e., revocation) are effective upon the next request for access (DA3.11). That is, the change affects all future send and receive operations, except if a process has already made a request for the message queue and is waiting for its availability (e.g., a process is waiting to receive a message), in which case the access change is not effective for that process until the next request (DA3.12).

If a process requests deletion of a message queue, it is not deleted until the last process that is waiting for the message queue receives its message (or equivalently, the last process waiting for a message in the queue terminates) (DA3.13). However, once a message queue has been marked as deleted, additional processes cannot perform messaging operations and it cannot be undeleted (DA3.14).

The default access control on newly created message queues is determined by the effective user ID and group ID of the process that created the message queue and the specific permissions requested by the process creating the message queue.

- The owning user and creating user of a newly created message queue will be the effective user ID of the creating process.
- The owning group and creating group of a newly created message queue will be the effective group ID of the creating process.
- The initial access permissions on the message queue must be specified by the creating process, or they are set to null and the object is inaccessible until the owner sets them.
- Message queues do not use ACLs as described above, they only have permission bits. (DA3.15)

Access permissions can be changed by any process with an effective user ID equal to the owning user ID or creating user ID of the message queue. Access permissions can also be changed by any process with an effective user ID of 0 (DA3.16).

6.2.3.3.3 DAC: Semaphores

For semaphores, access checks are performed for each access request (e.g., to lock or unlock the semaphore) (DA3.17). Changes to access controls (i.e., revocation) are effective upon the next request for access (DA3.18). That is, the change affects all future semaphore operations, except if a process has already made a request for the semaphore and is waiting for its availability, in which case the access change is not effective for that process until the next request (DA3.19).

In cases where an administrative user determines that immediate revocation of access to a semaphore is required, the administrative user can reboot the computer, thus destroying the semaphore and any processes waiting for it. This method is the described in the Security Guide. Since a semaphore exists only within a single host in the network, rebooting the particular host where the semaphores is present is sufficient to revoke all access to that semaphore.

If a process requests deletion of a semaphore, it is not deleted until the last process that is waiting for the semaphore obtains its lock (or equivalently, the last process waiting for the semaphore terminates) (DA3.20). However, once a semaphore has been marked as deleted, additional processes cannot perform semaphore operations and it cannot be undeleted (DA3.21).

The default access control on newly created semaphores is determined by the effective user ID and group ID of the process that created the semaphore and the specific permissions requested by the process creating the semaphore (DA3.22).

- The owning user and creating user of a newly created semaphore will be the effective user ID of the creating process.
- The owning group and creating group of a newly created semaphore will be the effective group ID of the creating process.
- The initial access permissions on the semaphore must be specified by the creating process, or they are set to null and the object is inaccessible until the owner sets them.
- Semaphores do not have ACLs as described above, they only have permission bits (DA3.23).

Access permissions can be changed by any process with an effective user ID equal to the owning user ID or creating user ID of the semaphore (DA3.24). Access permissions can also be changed by any process with an effective user ID of 0 (DA3.25).

This function contributes to satisfy the security requirement FDP_ACC.1, FDP_ACF.1, FMT_MSA.1, FMT_SMF.1, FMT_MSA.3.

6.2.4 Object Reuse (OR)

Object Reuse is the mechanism that protects against scavenging, or being able to read information that is left over from a previous subject's actions. Explicit initialization is appropriate for most TSF-managed abstractions, where the resource is implemented by some TSF internal data structure whose contents are not visible outside the TSF: queues, datagrams, pipes, and devices. These resources are completely initialized when created, and have no information contents remaining.

Explicit clearing is used in SLES only for directory entries, because they are accessible in two ways: through TSF interfaces both for managing directories and for reading files. Because this exposes the internal structure of the resource, it must be explicitly cleared on release to prevent the internal state from remaining visible.

Storage management is used in conjunction with explicit initialization for object reuse on files, and processes. This technique keeps track of how storage is used, and whether it can safely be made available to a subject.

The following sections describe in detail how object reuse is handled for the different types of objects and data areas and how the requirements defined in FDP_RIP.2 are satisfied.

6.2.4.1 Object Reuse: File System Objects (OR.1)

All file system objects (FSOs) available to general users are accessed by a common mechanism for allocating disk storage and a common mechanism for paging data to and from disk. This includes the Journaling File System (ext3).

Object reuse is irrelevant for the CD-ROM File System (ISO-9660) because it is a read-only file system and so it is not possible for a user to read residual data left by a previous user. File systems on other media (tapes, diskettes.) are irrelevant because of warnings in the Security Guide not to mount file systems on these devices.

For this analysis, the term FSO refers not only to named file system objects (files, directories, device special files, named pipes, and UNIX domain sockets) but also to other abstractions that use file system storage (symbolic links and unnamed pipes). All of these, except unnamed pipes, have a directory entry that contains the last part of the pathname and an inode that controls access rights and points to the disk blocks used by the FSO.

In general, file system objects are created with no contents, directories and symbolic links are exceptions, and some of their content is specified at creation time (OR1.1).

This function contributes to satisfy the security requirement FDP_RIP.2.

6.2.4.2 Object Reuse: IPC Objects (OR.2)

SLES shared memory, message queues, and semaphores are initialized to all zeroes at creation. These objects are of a finite size (shared memory segment is from one byte to the value defined in /proc/sys/kernel/shmmax, semaphore is one bit), and so there is no way to grow the object beyond its initial size (OR2.1).

No processing is performed when the objects are accessed or when the objects are released back to the pool.

This function contributes to satisfy the security requirement FDP_RIP.2.

6.2.4.3 Object Reuse: Memory Objects (OR.3)

A new process's context is completely initialized from the process's parent when the fork system call is issued. All program visible aspects of the process context are fully initialized. All kernel data structures associated with the new process are copied from the parent process, then modified to describe the new process, and are fully initialized (OR3.1).

The Linux kernel zeroes each memory page before allocating it to a process. This pertains to memory in the program's data segment and memory in shared memory segments (OR3.2). When a process requests more memory from the kernel, the memory is explicitly cleared before the process can gain access to it (OR3.3). This does not include memory that has been buffered by the library routines used by process. But this memory has already been allocated to the process by the kernel (cleared for object reuse at that time). Note that process internal memory management and buffering is not subject of this Security Target.

When the kernel performs a context switch from one thread to another, it saves the previous thread's General Purpose Registers (GPRs) and restores the new thread's GPRs, completely overwriting any residual data left in the previous thread's registers (OR3.4). Floating Point Registers (FPRs) are saved only if a process has used them. The act of accessing an FPR causes the kernel to subsequently save and restore all the FPRs for the process, thus overwriting any residual data in those registers (OR3.5).

Processes are created with all attributes taken from the parent. The process inherits its memory (text and data segments), registers, and file descriptors from its parent (OR3.6). When a process execs a new program, the text segment is replaced entirely.

This function contributes to satisfy the security requirement FDP_RIP.2.

6.2.5 Security Management (SM)

This section describes the functions for the management of security attributes that exist within SLES.

6.2.5.1 Roles (SM.1)

A simple role model is used for this evaluation that just supports two roles: administrative users and normal users (SM1.1).

In the evaluated confguration a user has the role of an administrative when he is allowed to *su* to root. Root itself will not be used as a userid where a user can directly log in to (except for login from the system console). So every administrative user has his/her own userid, which is used to log into the system.

6.2.5.1.1 Administrative Users

Users that are allowed to *su* to root can perform administrative actions (provided they also know the password required to su to root). Users that don't have the privilege to use *su* in their user profile can not perform administrative actions even if they know the root password (SM1.2).

6.2.5.1.2 Normal Users

Normal users can not perform actions that require root privileges. They can only execute those setuid root programs they have access to (SM1.3). In the evaluated configuration this is restricted to those programs they need like the passwd program that allows a user to change his/her own password.

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

6.2.5.2 Access Control Configuration and Management (SM.2)

Acess control to objects is defined by the permission bits or by the Access Control Lists (for those objects that have access control lists asscociated with them). Default access permission bits are defined in the system configuration files that define the value of the access control bits for objects being created without explicit definition of the permission bits. The administrative user can define and modify those default values.

Permissions can be changed by the object owner and an administrative user (SM2.1). When an object is created the creator is the object owner (SM2.2). Object ownership can be transferred (SM2.3). In the case of IPC objects, the creator will always have the same right as the owner, even when the ownership has been transferred (SM2.4).

This function contributes to satisfy the security requirements FMT_MSA.1, FMT_MSA.3, FMT_SMF.1 and FMT_REV.1 „Object Attributes".

6.2.5.3 Management of User, Group and Authentication Data (SM.3)

6.2.5.3.1 Creating new Users

An administrative user can create a new user and assigns a unique userid to this user. The initial password has to be defined using the *passwd* command. The new user will be disabled until the initial password is set (SM3.1).

Attributes that can be set for each user are among others (a complete list can be found in the description of the *useradd* command and the description of the content of the file */etc/passwd*):

- Administrative status of the user
- List of groups the user belongs to
- Home directory for this user

Those attributes are stored in the file */etc/passwd*. (SM3.2)

6.2.5.3.2 Modification of user attributes

User attributes can be modified by an administrative user. Modifications of user attributes require the modification of the administration database that contains the user attributes (mainly */etc/shadow*) (SM3.3).

6.2.5.3.3 Management of Authentication Data

An administrative user has the capability to define rules and restrictions for passwords used to authenticate users. The parameter available are:

- The number of days (since January 1, 1970) since the password was last changed.
- The number of days before password may be changed (0 indicates it may be changed at any time)
- The number of days after which password must be changed (99999 indicates user can keep his or her password unchanged for many, many years)
- The number of days to warn user of an expiring password (7 for a full week)

- The number of days after password expires that account is disabled (SM3.4)

All users are also allowed to change their own password using the *passwd* command. The password restrictions defined by the administrative user apply (SM3.5).

This list of attributes satisfies those required by FIA_ATD.1. In addition this function contributes to satisfy the security requirements FIA_SOS.1, FMT_MTD.1 „User Attributes", FMT_MTD.1 „Authentication Data", FMT_SMF.1 and FMT_REV.1 „User Attributes".

6.2.6 TSF Protection (TP)

While in operation, the kernel software and data are protected by the hardware memory protection mechanisms described in the high level design and the hardware reference manuals for the underlying hardware (Intel Pentium 4 and Xeon processors). The memory and process management components of the kernel ensure a user process cannot access kernel storage or storage belonging to other processes (TP1.1).

Non-kernel TSF software and data are protected by DAC and process isolation mechanisms. In the evaluated configuration, the reserved user ID root, or other reserved IDs equivalent to root, owns TSF directories and files In general, files and directories containing internal TSF data (e.g. batch job queues) are also protected from reading by DAC permissions (TP1.2).

The TSF and the hardware and firmware components are required to be physically protected from unauthorized access. The system kernel mediates all access to the hardware mechanisms themselves, other than program visible CPU instruction functions.

The boot image for each host with the evaluated TOE in the networked system is adequately protected.

6.2.6.1 TSF Invocation Guarantees (TP.1)

All system protected resources are managed by the TSF. Because all TSF data structures are protected, these resources can be directly manipulated only by the TSF, through defined TSF interfaces. This satisfies the condition that the TSF must be "always invoked" to manipulate protected resources (TP1.3).

Resources managed by the kernel software can only be manipulated while running in kernel mode (TP1.4).

Processes run in user mode and can call functions of the kernel only as the result of an exception or interrupt (TP1.5). The hardware and the kernel software handling these events and ensure that the kernel is entered only at pre-determined locations, and within pre-determined parameters. All kernel managed resources are protected such that only the kernel software is able to manipulate them.

Trusted processes implement resources managed outside the kernel. The trusted processes and the data defining the resources are protected as described above depending on the type of interface. For directly invoked trusted processes the program invocation mechanism ensures that the trusted process always starts in a protected environment at a predetermined point (TP1.6). Other trusted process interfaces are started during system initialization and use well defined protocol or file system mechanisms to receive requests (TP1.7).

Some system calls or parameter of system calls are reserved are reserved for trusted processes. When called the kernel checks that the calling process runs with an effective userid of 0 (TP1.8).

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

6.2.6.2 Kernel (TP.2)

The SLES software consists of a privileged kernel and a variety of non-kernel components (trusted processes). The kernel operates on behalf of all processes (subjects).

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

Upon entry the kernel determines the function to be performed, performs it, and, when finished, performs another context switch to return to user processing (eventually on behalf of a different subject) (TP2.1).

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

Because the kernel is shared among all processes, any process running "in the kernel" (that is, running in privileged hardware state as the result of a context switch) is able to directly reference the data structures that implement shared resources.

The major components of the kernel are memory management, process management, the file system, the system call interface, and the device drivers.

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

6.2.6.3 Kernel Modules (TP.3)

SLES supports dynamically loadable kernel modules that are loaded automatically on demand. Kernel modules are actually a part of the kernel that is not resident but loaded as part of the kernel when needed (TP3.1). Whenever a program wants the kernel to use a feature that is only available as a loadable module, and if the kernel hasn't got the module installed yet, the kernel will take care of the situation and make the best of it (TP3.2).

This is what happens:

- The kernel notices that a feature is requested that is not resident in the kernel.
- The kernel uses modprobe to load a module that fits this symbolic description.
- modprobe looks into its internal "alias" translation table to see if there is a match. This table can be reconfigured and expanded by having "alias" lines in "/etc/modules.conf".
- insmod is then asked to insert the module(s) that modprobe has decided that the kernel needs. Every module will be configured according to the "options" lines in "/etc/modules.conf".
- modprobe exits and tells the kernel that the request succeeded (or failed...)
- The kernel uses the freshly installed feature just as if it had been configured into the kernel as a "resident" part. (TP3.3)

In the TOE Kernel modules will be not be automatically removed from the kernel when they have not been used for a period of time. Removing them from the kernel needs to be done explicitely.

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

6.2.6.4 Trusted Processes (TP.4)

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

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

Trusted processes have all the kernel interfaces available for their use, but are limited to kernel-provided mechanisms for communication and data sharing, such as files for data storage and pipes, sockets and signals for communication.

The major functions implemented with trusted processes include user login, identification and authentication, batch processing, some network operations, system initialization, and system administration.

The kernel will check for each system call that requires root privileges if the process that issued the call has those privileges (TP4.1). If not, the kernel will refuse to perform the system call. The kernel will also for each access to an object protected by the any of DAC mechanism check, if the process has the required access rights for the attempted type of access.

Any program executed with root privileges has the ability to perform the actions of a trusted process. It is therefore important that a site operating a SLES system strictly controls those programs and prohibits that those programs are

modified or that programs from untrusted sources are executed with root privileges (TP4.2).

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

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

This function also contributes to FPT_SEP.1.

Note: Trusted processes may use system management commands or system calls as mentioned in the section on supporting functions that are not part of the TSF. But in any case the kernel will verify that the process has the right to perform the system call with the parameter specified by the caller and has the right to access all files with the intended access mode.

6.2.6.5 TSF Databases (TP.5)

Table 6-4 identifies the primary TSF databases used in SLES and their purpose. These are listed both as individual files (by pathname) or collections of files.

With the exception of databases listed with the User attribute (which indicates that a user can read, but not write, the file), all of these databases shall only be accessible to administrative users. None of these databases shall be modifiable by a user other than an administrative user.

Those databases are part of the file system and therefore the file system protection mechanisms of the TOE have to be used to protect those databases from unauthorized access. It is the task of the persons responsible for setting up and administrating the system to ensure that the access control features of the TOE are used throughout the lifetime of the system to protect those databases.

Each host system within the TOE maintains its own TSF database. Syncronizing those databases is not performed in the evaluated configuration. If such syncronization is required by an organization it is the responsibility of an administrative users of the TOE to achieve this either manually or with some automated assisitance.

Table 6-4 . Administrative Databases. This table lists other administrative files used to configure the TSF.

Database	Purpose
/etc/at.allow	Defines users allowed to use the at command
/etc/at.deny	Defines users not allowed to use at command. Checked, if /etc/at.allow does not exist. If exists and empty and no "allow" file exists, all users are allowed to use the at command
/etc/cron.d/*	contains programs to be scheduled by the cron daemon
/etc/cron.{weekly hourly daily monthly}/*	contains programs to be scheduled by the cron daemon on a weekly, hourly, daily or monthly schedule
/etc/crontab	commands to be scheduled by the cron daemon
/etc/ftpusers	contains users not allowed to use the ftp command
/etc/group	Stores group names, supplemental GIDs, and group members for all system groups.
/etc/gshadow	Stores group passwords and group administrator information
/etc/hosts	Contains hostnames and their address for hosts in the network. This file is used to resolve a hostname into an Internet address in the absence of a domain name server
/etc/inittab	Describes the process started by init program at different run levels
/etc/init.d/*	System startup scripts
/etc/ld.so.conf	File containing a list of colon, space, tab, newline, or comma spearated directories in which to search for libraries for run-time link bindings
/etc/login.defs	Defines various configuration options for the login process
/etc/modules.conf	This file links kernel internal device identifiers with kernel modules (regular files). In addition it contains possible configurations options for the various modules.

/etc/pam.d/*	This directory contains the configuration of PAM. In it there is one configuration for each application that performs identification and authorization. Each of the configuration file contains the PAM modules that are to be used for this procedure.
/etc/passwd	Stores user names, user IDs, primary group ID, user real name, home directory, shell for all system users.
/etc/securetty	Contains device names of tty lines on which root is allowed to login
/etc/security/pam_pwcheck.conf	Contains rules for to enforce the password strength policy
/etc/security/pam_unix2.conf	Contains configuration parameter for the pam_unix2 password authentication module
/etc/shadow	Defines user passwords in one-way encrypted form, plus additional characteristics
/etc/ssh/sshd_config	Contains ssh configuration parameter for the ssh server
/etc/sysconfig/*	Directory containing several configuration files for network services
/etc/vsftpd.conf	Contains configuration parameter for the vsftp server
/etc/xinetd.conf	Configuration parameter for the xinet daemon
/usr/lib/cracklib_dict.*	Contains the dictionary used by the cracklib pam module as part of the password strength policy
/var/log/faillog	Stores time and date of last failed login attempts for each user
/var/log/lastlog	Stores time and date of last successful for each user.
/var/spool/atjobs	Directory to store jobs scheduled by the at daemon
/var/spool/cron/tabs/root	Crontab file for the root user
/var/spool/cron/allow	File containing users allowed to use crontab
/var/spool/cron/deny	File containing users not allowed to use crontab. Evaluated only if no /var/spool/cron/allow exists. If exists and empty and no "allow" file exists, all users are allowed to use crontab.

These tables are not functions but they are part of the management of the TSF. As such they contribute to the system management security functional requirements FMT_MSA.3 and FMT_MTD.1 (User Attributes and Authentication Data) as well as FMT_SMF.1.

6.2.6.6 Internal TOE Protection Mechanisms (TP.6)

All kernel software has access to all of memory, and the ability to execute all instructions. In general, however, only memory containing kernel data structures is manipulated by kernel software. Parameters are copied to and from process storage (i.e., that accessible outside the kernel) by explicit internal mechanisms, and those interfaces only refer to storage belonging to the process that invoked the kernel (e.g., by a system call). Functions implemented in trusted processes are more strongly isolated than the kernel. Because there is no explicit sharing of data, as there is in the kernel address space, all communications and interactions between trusted processes take place explicitly through files and similar mechanisms.

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

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

6.3 Supporting functions not part of the TSF

6.3.1 System Management Tools

The administrative user can use the commands provided by SLES for system management activities. Those commands are seen as part of the system management tools.

This function contributes to satisfy the security requirements associated with the management of security attributes.

Note: System management tools and commands do not enforce any part of the TOE security policy. They just provide the tools for the administrative user to perform his administrative functions. The TSF still check that the caller is allowed to invoke the system calls used by those tools and checks that the caller has the required access rights to the objects (like configuration files) he is going to access.

6.3.2 User Processes

The SLES TSF primarily exists to support the activities of user processes. A user, or non-TSF, process has no special privileges or security attributes. The user process is isolated from interference by other user processes primarily through the CPU execution state and address protection mechanisms and the way they are used by the kernel, and also through the protections on TSF interfaces for process and file manipulation.

User processes are by definition untrusted and therefore do not contribute to any security function. The TSF ensure that user processes are encapsulated in such a way that they are separated from the TSF and from processes (trusted and untrusted) running with different attributes and will only be able to communicate with them using the defined TSF interfaces. User processes therefore do not contribute to any security function of the TOE.

6.4 Assurance Measures

The following table provides an overview, how the assurance measures of EAL2 and ALC_FLR.1 are met by SLES.

Table 6-5: Mapping Assurance Requirements to Documentation

Assurance Component	Documentation describing how the requirements are met
ACM_CAP.2	Configuration management procedures within SuSE are highly automated using a process supported by the AutoBuild tool.
ADO_DEL.1	SLES is delivered on CD / DVD in shrinkwrapped package to the customer. SuSE verifies the integrity of the production CDs / DVDs by checking a production sample. The certification-sles-eal2.rpm package as well as other packages that contain fixes must be downloaded from the SuSE web site. Since those packages are digtally signed the user is able and has to verify the integrity and authenticity of those packages.
ADO_IGS.1	Guidance for installation and sytem configuration is provided in the Security Guide.
ADV_FSP.1	The functional specification for SLES consists of the man pages that describe the system calls, the trusted commands as well as a description of the security relevant configuration files. A spreadsheet provided by the sponsor lists all system calls, trusted commands and security relevant configuration files with a mapping to their description in the overall documentation.
ADV_HLD.1	A high level design of the security functions of SLES is provided. This document provides an overview of the implementation of the security functions within the subsystems of SLES and points to other existing documents for further details where appropriate.
ADV_RCR.1	The correspondence information is provided as part of the functional specification (with the spreadsheet). An additional document providing the correspondence to the TOE Summary Specification has been provided to the evaluation facility.
AGD_ADM.1	The Security Guide and the SLES admin handbook plus a special README file contain the specifics for the secure administration of the evaluated configuration.
AGD_USR.1	The Security Guide and the SLES user handbook plus a special README file contain the specifics for the secure usage of the evaluated configuration.
ALC_FLR.1	The defect handling procedure SuSE has in place for the development of SLES requires to describe the defect with its effects, security implications, fixes and required verification steps.
ATE_COV.1	Detailed test plans are produced to test the functions of SLES. Those test plan include an analysis of the test coverage, an analysis of the functional interfaces tested and an analysis of the testing against the high level design.
ATE_FUN.1	Testing has been performed on the platforms that are defined in the Security Target. Test results are documented such that the tests can be repeated.
ATE_IND.2	All the required resources to perform their own tests are provided to the evaluation facility to perform their test. The evaluation facility has performed and documented the tests they have created and performed as part of the evaluation technical report for testing.
AVA_SOF.1	The Strength of Function Analysis has been provided for the mechanism based on

Assurance Component	Documentation describing how the requirements are met
	permutational or probabilistic algorithms as part of the developer's vulnerability analysis document.
AVA_VLA.1	A vulnerability analysis has been provided that describes the sponsor's approach to identify vulnerabilities of SLES as well as the results of the findings.

6.5 TOE Security Functions requiring a Strength of Function

The TOE has one security function (IA) that is implemented by a probabilistic or permutational mechanism. This is the authentication mechanism that uses passwords for user authentication. The strength claimed for this function is SOF-basic.

7 Protection Profile Claims

No claim for compliance with an existing Protection Profile is made.

8 Rationale

The rationale section provides additional information and demonstrates that the security objectives and the security functions defined in the previous chapter are consistent and sufficient to counter the threats defined in chapter 2.

8.1 Security Objectives Rationale

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

8.1.1 Security Objectives Coverage

Table 8-1: Mapping Objectives to threats , assumptions and policies

Objective	**Threat / Policy**
O.AUTHORIZATION	T.UAUSER, P.AUTHORIZED_USERS
O.DISCRETIONARY_ACCESS	T.UAACCESS, P.NEED_TO_KNOW
O.RESIDUAL_INFO	P.NEED_TO_KNOW, T.UAACCESS
O.MANAGE	P.AUTHORIZED_USERS, P.NEED_TO_KNOW, T.UAUSER,
O.ENFORCEMENT	P.AUTHORIZED_USERS, P.NEED_TO_KNOW

Table 8-2: Mapping objectives for the environment to threats, assumptions and policies

Env. Objective	**Threat / Assumption / Policy**
OE.ADMIN	A.MANAGE, A.NO_EVIL_ADMIN
OE.CREDEN	A.COOP
OE.INSTALL	TE.COR_FILE, A.MANAGE, A.NO_EVIL_ADMIN, A.PEER, A.NET_COMP
OE.PHYSICAL	A.LOCATE, A.PROTECT, A.CONNECT
OE.INFO_PROTECT	TE.COR_FILE, A.PROTECT, A.UTRAIN, A.UTRUST
OE.MAINTENANCE	TE.HWMF
OE.RECOVER	TE.HWMF, TE.COR_FILE
OE.SOFTWARE_IN	P.NEED_TO_KNOW
OE.SERIAL_LOGIN	A.CONNECT
OE.PROTECT	TE.COR_FILE, A.NET_COMP, A.CONNECT
OE.HW_SEP	TE.HW_SEP

Table 8-3: Mapping threats to objectives

Threat	**Objective**
T.UAUSER	O.AUTHORIZATION, O.MANAGE
T.UAACCESS	O.DISCRETIONARY_ACCESS, O.RESIDUAL_INFO
TE.HWMF	OE.MAINTENANCE, OE.RECOVER
TE.COR_FILE	OE.PROTECT, OE.INSTALL, OE.INFO_PROTECT, OE.RECOVER
TE.HW_SEP	OE.HW_SEP

Table 8-4: Mapping Assumptions to Objectives

Assumption	**Objective**
A.LOCATE	OE.PHYSICAL
A.PROTECT	OE.INFO_PROTECT, OE.PHYSICAL
A.MANAGE	OE.ADMIN, OE.INSTALL, OE.RECOVER
A.NO_EVIL_ADMIN	OE.ADMIN, OE.INSTALL
A.COOP	OE.CREDEN
A.UTRAIN	OE.INFO_PROTECT
A.UTRUST	OE.INFO_PROTECT

A.NET_COMP	OE.PROTECT, OE.INSTALL
A.PEER	OE.INSTALL
A.CONNECT	OE.SERIAL_LOGIN, OE.PROTECT, OE.PHYSICAL

Table 8-5: Mapping Policies to Objectives

Policy	**Objective**
P.AUTHORIZED_USERS	O.AUTHORIZATION, O.MANAGE, O.ENFORCEMENT
P.NEED_TO_KNOW	O.DISCRETIONARY_ACCESS, O.MANAGE, O.ENFORCEMENT, O.RESIDUAL_INFO, OE.SOFTWARE_IN

8.1.2 Security Objectives Sufficiency

T.UAUSER: The threat of impersonization of an authorized user by an attacker is sufficiently diminished by O.AUTHORIZATION requiring proper authorization of users gaining access to the TOE. O.MANAGE ensures that only administrative users (which are assumed to be trustworthy) have the ability to add new users or modify the attributes of users. Together those objectives ensure that no unauthorized user can impersonate as an authorized user.

T.UAACCESS: The threat of an authorized user of the TOE accessing information resources without the permission from the user responsible for the resource is removed by O.DISCRETIONARY_ACCESS requiring access control for resources and the ability for authorized users to specify the access to their resources. This ensures that a user can access a resource only if the requested type of access has been granted by the user responsible for the management of access rights to the resource. In addition O.RESIDUAL_INFO ensures that an authorized user can not gain access to the information contained in a resource after the resource has been released to the system for reuse.

TE.HWMF: The threat of losing data due to hardware malfunction is mitigated by OE.MAINTENANCE requiring the invocation of diagnostic tools during preventative maintenance periods. In addition OE.RECOVER requires the organizational procedures to be set up that are able to recover critical data and restart operation in a secure mode in the case such a hardware malfunction happens.

TE.COR_FILE: The threat of undetected loss of integrity of security enforcing or relevant files of the TOE is diminished by OE.INSTALL requiring procedures for secure distribution, installation and configuration of systems therby ensuring that the system has a secure inital state with the required protection of such files, OE.PROTECT requiring protection of transferred data in the network the TOE is connected to and OE.INFO_PROTECT requiring procedures for the appropriate definition of access rights to protect those files when the system is up and running. OE.RECOVER ensures that the system is securely recovered, which includes the verification of the integrity of security enforcing or security relevant files as part of the recovery procedures.

TE.HW_SEP: The threat that the underlying hardware does not provide the functions required to implement an efficient self-protection of the TSF such that the TSF themselves and the TSF data can be efficiently protected from unauthorized access and modification by untrusted software is addressed by the objective OE.HW_SEP for the processor used to execute the TOE software. This is a basic fundamental requirement for secure operating systems where trusted and untrusted software are executed on the same processor using the same memory space and the same processor resources. For TSF self-protection a processor feature is required that controls access to processor resources and main memory such that the TSF can implement a self-protection function in the way that the TSF reserve processor resources and memory areas for themselves and prohibit that those resources can be used by non-TSF software.

A.LOCATE: The assumption on physical protection of the processing resources of the TOE is covered by OE.PHYSICAL requiring physical protection.

A.PROTECT: The assumption on physical protection of all hard- and software as well as the network and peripheral cabling is covered by the objectives OE.INFO_PROTECT demanding the approval of network and peripheral cabling and OE.PHYSICAL requiring physical protection.

Note: Phsical protection of the network components and cabling is required by A.PROTECT which may seem to be redundant to A.CONNECT. But A.CONNECT also addresses protection against passive wiretapping, which may be done without having physical access to a hardware component.

A.MANAGE: The assumption on competent administrators is covered by OE.ADMIN requiring competent and trustworthy administrators and OE.INSTALL requiring procedures for secure distribution, installation and configuration of systems as well as OE.RECOVER requiring the administrator to perform all the required actions to bring the TOE into a secure state after a system failure or discontinuity..

A.NO_EVIL_ADMIN: The assumption on administrators that are neither careless nor wilfully negligent or hostile is covered by OE.ADMIN requiring competent and trustworthy administrators and OE.INSTALL requiring procedures for secure distribution, installation and configuration of systems.

A.COOP: The assumption on authorized users to act in a cooperating manner is covered by the ojective OE.CREDEN requiring the safe storage and non-disclosure of authentication credentials.

A.NET_COMP: The assumption on network components to not modify transmitted data is covered by the objective OE.PROTECT requiring procedures and/or mechanisms to ensure a safe data transfer between systems as well as OE.INSTALL requiring proper installation and configuration of all parts of the networked system thus including also components that are not part of the TOE.

A.PEER: The assumption on the same management control and security policy constraints for systems with which the TOE communicates is covered by OE.INSTALL requiring procedures for secure distribution, installation and configuration of the networked system.

A.CONNECT: The assumption on controlled access to peripheral devices and protected internal communication paths is covered by OE.SERIAL_LOGIN for the protection of attached serial login devices, OE.PROTECT for the protection of data transferred between workstations and OE.PHYSICAL requiring physical protection.

A.UTRAIN: The assumption on trained users is covered by OE.INFO_PROTECT which requires that users are trained to protect the data belonging to them.

A.UTRUST: The assumption on user to be trusted to protect data is covered by OE.INFO_PROTECT which requires that users are trusted to use the protection mechanims of the TOE adequately to protect their data.

P.AUTHORIZED_USERS: The policy demanding that users have to be authorized for access to the system is implemented by O.AUTHORIZATIONand supported by O.MANAGE allowing the management of this functions and O.ENFORCEMENT ensuring the correct invocation of the functions.

P.NEED_TO_KNOW: The policy to restrict access to and modification of information to authorized users which have a „need to know" for that information is implemented by O.DISCRETIONARY_ACCESS demanding an appropriate access control function that allows to define access rights down to the granularity of an individual user. It is supported by O.RESIDUAL_INFO ensuring that resources do not release such information during reuse and by OE.SOFTWARE_IN preventing users other than administrative users from installing new software that might affect the access control functionality. O.MANAGE allows administrative and normal users (for the files they own) to manage these functions,
O. ENFORCEMENT ensures that the functions are invoked and operate correctly.

8.2 Security Requirements Rationale

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

8.2.1 Internal Consistency of Requirements

This section describes the mutual support and internal consistency of the components selected for this Security Target. These properties are discussed for both functional and assurance components.

The functional components were selected from CC components defined in part 2 of the Common Criteria. Functional component FMT_SMF.1 (Specification of Management Functions) has been added in accordance with AIS 32, Final Interpretation 065. The use of component refinement was accomplished in accordance with CC guidelines.

Multiple instantiation of identical or hierarchically-related components was used to clearly state the required functionality that exists in this Security Target.

For internal consistency of the requirements we provide the following rationale:

Discretionary Access Control

FDP_ACC.1 requires the existence of a Discretionary Access Control Policy for file system objects and and Inter Process Communication objects. The rules of this policy are described in FDP_ACF.1. Management of access rights is defined in FMT_MSA.1 and FMT_REV.1. To be effective a discretionary access control mechanism requires user's to be properly identified and authenticated (as required by FIA_UID.2 and FIA_UAU.2), proper binding of subjects to

users (as required by FIA_USB.1), reference mediation (as required by FPT_RVM.1) and domain separation (as required by FPT_SEP.1). The policy is also supported by the requirement for residual information protection (FDP_RIP.2) which prohibits that users access information they are not authorized to via residuals remaining in objects that the allocate.

Identification and Authentication

As stated above Identification and Authentication is required for a useful discretionary access control based on the identity of individual users. FIA_UAU.2 and FIA_UID.2 require that users are authenticated before they can perform any action on the TOE. FIA_SOS.1 ensures that the mechanism used for authentication (passwords) has a minimum strength and FIA_UAU.7 provides some level of protection against simple spoofing in the TOE environment. Since the TOE implements processes acting on behalf of the user FIA_USB.1 ensures that those processes act within the limits defined for the user they are acting for (unless they are trusted to perform activities beyond the rights of the user).

Object Reuse

As stated above object resuse (as required by FDP_RIP.2) is a supporting function that prohibits easy access to information via residuals left in objects when they are re-allocated to another user. As this the function supports the intention of the discretionary access control policy.

Security Management

The functions defined so far require several management functions as defined by FMT_SMF.1.

The first one is the management of access rights (as defined by FMT_MSA.1 and FMT_REV.1 "Revocation of Object Attributes"). In addition new objects require to have default access rights which are required by FMT_MSA.3.

The second one is the management of users, which is defined in FMT_MTD.1 "Management of User Attributes" and FMT_REV.1 " Revocation of User Attributes". Since passwords are used for authentication the management of this authentication data is also required in FMT_MTD.1 "Management of Authentication Data". In addition the TOE supports two roles (administrative user, which is equal to root and normal user) which is expressed by FMT_SMR.1

TSF Protection

The TOE needs to ensure that users are limited in their activities by the boundaries defined by the access control policy. To ensure this the TSF need to check all access of users to protected objects (as required by FPT_RVM.1) and maintain a domain for its own execution that protects it from inference and tampering by any subject that is not part of the TSF. This is expressed with the requirement FPT_SEP.1.

The following table shows how the security functional requirements map to the objectives defined for the TOE.

Table 8-6: Mapping Objectives to Security Functional Requirements

O.AUTHORIZATION	User Attribute Definition (FIA_ATD.1) Strength of Authentication Data (FIA_SOS.1) Authentication (FIA_UAU.2) Protected Authentication Feedback (FIA_UAU.7) Identification (FIA_UID.2) User-Subject Binding (FIA_USB.1) Management of Authentication Data (FMT_MTD.1)
O.DISCRETIONARY_ACCESS	Discretionary Access Control Policy (FDP_ACC.1) Discretionary Access Control Functions (FDP_ACF.1) User Attribute Definition (FIA_ATD.1) User-Subject Binding (FIA_USB.1) Management of Object Security Attributes (FMT_MSA.1) Static Attribute Initialization (FMT_MSA.3) Revocation of Object Attributes (FMT_REV.1)
O.RESIDUAL_INFO	Object Residual Information Protection (FDP_RIP.2)
O.MANAGE	Management of Object Security Attributes (FMT_MSA.1) Static Attribute Initialization (FMT_MSA.3) Management of User Attributes (FMT_MTD.1) Management of Authentication Data (FMT_MTD.1) Revocation of User Attributes (FMT_REV.1) Revocation of Object attributes (FMT_REV.1) Specification of Management Functions (FMT_SMF.1)

	Security Management Roles (FMT_SMR.1)
O.ENFORCEMENT	Reference Mediation (FPT_RVM.1) Domain Separation (FPT_SEP.1)

O.AUTHORIZATION

The TSF must ensure that only authorized users gain access to the TOE and its resources. Users authorized to access the TOE have to use an identification and authentication process [FIA_UID.2, FIA_UAU.2]. To ensure authorized access to the TOE, authentication data is protected [FIA_ATD.1, FIA_UAU.7, FMT_MTD.1 "Management of Authentication Data"]. The strength of the authentication mechanism must be sufficient to ensure that unauthorized users can not easily impersonate an authorized user [FIA_SOS.1]. Proper authorization for subjects acting on behalf of users is also ensured [FIA_USB.1].

O.DISCRETIONARY_ACCESS

The TSF must control access to resources based on identity of users. The TSF must allow authorized users to specify which resources may be accessed by which users.

Discretionary access control must have a defined scope of control [FDP_ACC.1]. The rules of the DAC policy must be defined [FDP_ACF.1]. The security attributes of objects used to enforce the DAC policy must be defined. The security attributes of subjects used to enforce the DAC policy must be defined [FIA_ATD.1, FIA_USB.1]. Authorized users must be able to control who has access to objects [FMT_MSA.1] and be able to revoke that access [FMT_REV.1 "Revocation of Object Attributes"]. Protection of named objects must be continuous, starting from object creation [FMT_MSA.3].

O.RESIDUAL_INFORMATION

The TSF must ensure that any information contained in a protected resource is not released when the resource is recycled.

Residual information associated with defined objects in the TOE must be purged prior to the reuse of the object containing the residual information [FDP_RIP.2].

O.MANAGE

The TSF must provide all the functions and facilities necessary to support the administrative users that are responsible for the management of TOE security.

Aspects that need to be managed must be defined [FMT_SMF.1]. The TSF must provide for an administrative user to manage the TOE [FMT_SMR.1]. The administrative user must be able to administer user accounts [FMT_MTD.1 "Management of User Attributes", FMT_MTD.1 "Management of Authentication Data", FMT_REV.1 "Revocation of User Attributes"] and object attributes [FMT_MSA.1, FMT_REV.1 "Revocation of Object Attributes"]. In addition the default values for access control need to be defined [FMT_MSA.3].

O.ENFORCEMENT

The TSF must be designed and implemented in a manner which ensures that the organizational policies are enforced in the target environment.

The TSF must make and enforce the decisions of the TSP [FPT_RVM.1]. It must be protected from interference that would prevent it from performing its functions [FPT_SEP.1]. The correctness of this objective is further met through the assurance requirements defined in this Security Target.

This objective provides global support to other security objectives for the TOE by protecting the parts of the TOE which implement policies and ensures that policies are enforeced.

No security functions for the non-IT environment have beed added, since the procedures that need to be implemented can (and probably will) be different for each site running the evaluated version of SLES. Therefore no specific security functional requirements and security functions for the non-IT environment have been defined in this Security Target. Individual sites running SLES should validate that the procedures and physical security measures they have put in place are sufficient to cover the security objectives defined for the environment of the TOE in this Security Target.

Security requirements for the IT environment have been added to define the support required by the TOE from the underlying processor. As with every operating system that also runs untrusted software, some kind of separation mechanism must exists that prohibits the untrusted software from tampering with trusted software and TSF data. In the

case of this TOE the processor must supply a separation mechanism such that memory areas as well as hardware privileges required to directly access devices or memory management functions are protected from direct access by untrusted software. This is defined with an access control policy called „memory access control policy" that the underlying processor must support. This policy is expressed using FDP_ACC.1 and FDP_ACF.1 as well as FDP_MSA.3 from part 2 of the Common Criteria.

8.2.2 Security Requirements Instantiation Rationale

This section provides tha rationale for the selections and instantiations made in the security requirements section for the security requirements taken from part 2 of the Common Criteria.

In FDP_ACC.1 the different objects that SLS controls with a discretionary access control function are listed.

FDP_ACF.2 gets somewhat complicated with expressing the different policies for discretionary access control for the different types of objects. It was decided to list the rules for file system objects, IPC objects separately because they differ significantly.

In FIA_ATD.1 nothing have been added as additional security attribute of users within the evaluated configuration of SLES. Other attributes as for example stored in the file */etc/shadow* are not seen as security attributes.

In FIA_USB.1 the way how SLES associates the real and effective user ID is expressed. While the effective user id and group id can change as the result of a su command or a program with the setuid or setgid attribute set, the real and is maintained and allow to trace activities to the real user that originated them.

In FMT_REV.1 „Revocation of User Attributes" the delayed revocation method has been added, since this is the standard way SLES behaves. To get immediate revocation the administrative user has to force the user to log off after he has made the modifications to the users attribute.

In FMT_REV.1 „Revocation of Object Attributes" the SLES implementation of delayed revocation is defined.

FMT_SMF.1 has been added to comply with AIS 32, Final Interpretation 065 and the dependencies defined there. The Security Target defines management requirements in FMT_MSA.1 and the two instantiations of FMT_MTD.1 for

- User attribute management
- Authentication data management

those aspects are listed in this security functional requirement.

FMT_SMR.1 defines only the roles of administrative and normal users.

8.2.3 Security Requirements Coverage

The following table shows that each security functional requirement addresses at least one objective.

Table 8-7: Mapping Security Functional Requirements to Objectives

SFR	Objectives
FDP_ACC.1	O.DISCRETIONARY_ACCESS
FDP_ACF.1	O.DISCRETIONARY_ACCESS
FDP_RIP.2	O.RESIDUAL_INFO
FIA_ATD.1	O.AUTHORIZATION O.DISCRETIONARY_ACCESS
FIA_SOS.1	O.AUTHORIZATION
FIA_UAU.2	O.AUTHORIZATION
FIA_UAU.7	O.AUTHORIZATION
FIA_UID.2	O.AUTHORIZATION
FIA_USB.1	O.AUTHORIZATION, O.DISCRETIONARY_ACCESS
FMT_MSA.1	O.DISCRETIONARY_ACCESS O.MANAGE
FMT_MSA.3	O.DISCRETIONARY_ACCESS O.MANAGE
FMT_MTD.1	O.MANAGE

SFR	Objectives
User Attributes	
FMT_MTD.1 Authen. Data	O.AUTHORIZATION O.MANAGE
FMT_REV.1 User Attributes	O.DISCRETIONARY_ACCESS O.MANAGE
FMT_REV.1 Obj. Attributes	O.DISCRETIONARY_ACCESS O.MANAGE
FMT_SMF.1	O.MANAGE
FMT_SMR.1	O.MANAGE
FPT_RVM.1	O.ENFORCEMENT
FPT_SEP.1	O.ENFORCEMENT

8.2.4 Rationale for Security Requirements for the IT environment

Those requirements define the need for an access control policy implemented in the underlying processor that allows to reserve the access and manipulation of critical processor and memory resources to specially software (instructions) operating with a defined privilege attribute (usually called "supervisor" or "system" mode). The TSF have to ensure that no untrusted software will ever execute with this privilege. Based on this the TSF can then control the access to memory objects and other processor resources and implement the high level access control functions as well as the TSF self protection.

To do this the underlying processor has to provide a basic access control mechanism where access to processor resources (like registers) and memory areas is controlled based on a processor attribute where the implementation of the TSF ensure that untrusted software never executes with this attribute. This is expressed with FDP_ACC.1 and FDP_ACF.1. Since the processor may allow read access to specific registers for software running without „supervisor" privilege, FDP_ACF.1.3 is used to define this.

The requirements don't define the exact rules because those may differ slightly for different processor types without getting into the problem of interoperability problems. For example a new processor may implement additional instructions and additional register but still be fully downwards compatible. Since software developed for the older versions of the processor will not use the additonal instructions and will not touch the additional register, the claims for the software still hold although the objects controlled by the new processor differ from those controlled by the old processor. Of course, if anybody wants to evaluate the underlying processor those rules have to be defined precisely for the specific processor type that is the target of the hardware evaluation.

The "static attribute initialization" (FMT_MSA.3) is here defined as the value of the processor attribute ("user" or "supervisor") at the start-up of the processor (after reset or power-up). This has to be "permissive" since the register and memory areas need to be initialized. It is therefore necessary that the software that perform those initalization activities is part of the TSF.

The security requirements for the IT environment address the security objective OE.HW_SEP since the memory access control policy allows the TOE to protect the TSF and the TSF data from unauthorized access by untrusted software. The TOE has to use the memory access control policy to allow memory access by untrusted software just to those memory areas that belong to the untrusted software itself. Access to special hardware register will be managed by the TSF such that this access will always be reserved to trusted software. This shows that the security requirements for the IT environment are sufficient to protect the TSF and TSF data from unauthorized access and modification when used correctly by the TOE. The following table shows the mapping of the security functional requirements for the IT environment to the security objectives for the IT environment:

Table 8-8: Mapping Security Functional Requirements for the IT Environment to Objectives

SFR	Objective
FDP_ACC.1	OE.HW_SEP
FDP_ACF.1	OE.HW_SEP
FMT_MSA.3	OE:HW_SEP

8.2.5 Security Requirements Dependency Analysis

The following table shows the dependencies between the different security functional requirements and if they are resolved in this Security Target.

Table 8-9: Dependencies between Security Functional Requirements

Security Functional Requirement	Dependencies	Resolved
FDP_ACC.1	FDP_ACF.1 Security attribute based access control	yes
FDP_ACF.1	FDP_ACC.1 Subset access control FMT_MSA.3 Static attribute initialisation	yes
FDP_RIP.2	No dependencies.	yes
FIA_ATD.1	No dependencies	yes
FIA_SOS.1	No dependencies	yes
FIA_UAU.2	FIA_UID.1 Timing of identification	yes
FIA_UAU.7	FIA_UAU.1 Timing of authentication	yes
FIA_UID.2	No dependencies	yes
FIA_USB.1	FIA_ATD.1 User attribute definition	yes
FMT_MSA.1	[FDP_ACC.1 Subset access control or FDP_IFC.1 Subset information flow control] FMT_SMR.1 Security roles FMT_SMF.1Specification of management function	yes
FMT_MSA.3	FMT_MSA.1 Management of security attributes FMT_SMR.1 Security roles FMT_SMF.1 1Specification of management function	yes
FMT_MTD.1 User Attriutes	FMT_SMR.1 Security roles	yes
FMT_MTD.1 Authentication Data	FMT_SMR.1 Security roles	yes
FMT_REV.1 User Attributes	FMT_SMR.1 Security roles	yes
FMT_REV.1 Object Attributes	FMT_SMR.1 Security roles	yes
FMT_SMF.1	No dependencies	yes
FMT_SMR.1	FIA_UID.1 Timing of identification	yes
FPT_RVM.1	No dependencies	yes
FPT_SEP.1	No dependencies	yes

Some remarks:

The dependencies of FIA_UAU.2, FIA_UAU.7 and FMT_SMR.1 on FIA_UID.1 are resolved with the inclusion of FIA_UID.2 which is hierarchical to FIA_UID.1

The dependencies of FMT_MSA.1 and FMT_MSA.3 on FMT_SMF.1 were introduced by AIS 32, Final Interpretation 065 and have been considered here.

The multiple instantiations of FMT_MTD.1 and FMT_REV.1 have been included in this table, since a multiple instantiation of one security functional requirement may in some cases result in the requirement for multiple instantiations of depending requirements. This is not the case here, since they all rely on the same simple role model of the TOE.

This table shows that no unresolved dependencies exist between security functional requirements.

There are also no unresolved dependencies between security assurance requirements. This is because the evaluation assurance level EAL2 has been defined such that no unresolved dependencies exist. The additional assurance component ALC_FLR.1 has no dependencies and therefore there are no unreolved dependencies for assurance components.

8.2.6 Strength of function

This Security Target claims a SOF rating SOF-basic. This claim applies for FIA_SOS.1, whereby it is stated that a 'one off' probability of guessing the password in 1,000,000 is given. The SFR is in turn consistent with the security objectives. A claim of SOF-basic is also consistent with the assumption of a non-hostile user community and the assumption on physical protection which prohibits that well-skilled, hostile attackers get physical access to the TOE.

8.2.7 Evaluation Assurance Level

This security target claims EAL2 augmented with ALC_FLR.1, which is seen appropriate for a well-controlled, non-hostile environment.

8.3 TOE Summary Specification Rationale

8.3.1 Security Functions Justification

The following table shows that the IT security functions, as specified in the TOE summary specification, meet all security functional requirements for the TOE and work together to satisfy the TOE security functional requirements.

Table 8-10: Mapping Security Functional Requirements to Security Functions

SFR	**security functions (TOE summary specification)**
FDP_ACC.1	The discretionary access control policy is based on **DA** defining permission bits for the subjects and objects as there are file system objects and IPC objects.
FDP_ACF.1	The discretionary access control is realized as described above by **DA**. There the individual mechanisms for access control depending on the object type are described in detail.
FDP_RIP.2	Object residual information protection is realized by security functions for object reuse (**OR**) on file system objects, IPC objects, queing system objects and miscellaneous objects.
FIA_ATD.1	Security attributes belonging to individual users are realized by the user I&A data management of **IA**. Management of user attributes is described in **SM**.
FIA_SOS.1	The passwd funtion of **IA** is able to enforce the verification of secrets as required. System management commands can be used to define parameters that can be used to (hopefully) enhance the strength of the passwords chosen by the user. Password management including the possible parameter to enhance the strength of passwords are explained in **SM**.
FIA_UAU.2	Authentication of each user before any action is realized by **IA** (common authentication mechanism and interactive login and related mechanisms). Authentication is initated by a trusted process. Trusted processes are described in **TP**.
FIA_UAU.7	The login mechanisms of **IA** provide only obscured feedback during authentication. Authentication feedback is managed by a trusted process. Trusted processes are described in **TP**.
FIA_UID.2	Identification of each user before any action is realized together with authentication as in **IA** (see above). Identification is initated by a trusted process. Trusted processes are described in **TP**.
FIA_USB.1	The required binding between subjects and users is implemented by the su functionality of **IA** and login processing . There also the logoff process is described which releases the binding between subjects and users.
FMT_MSA.1	The management of object security attributes is implemented by the access control configuration and management function **SM**, the objects are described in **DA** (file system objects and IPC

SFR	security functions (TOE summary specification)
	objects).
FMT_MSA.3	Restrictive default values for security attributes are defined for the objects when they are created. Default values can be defined by an administrative user for all object types and by the user for file system objects created under his control. (see above, i.e. **SM** and **DA**). Some default values are defined in TSF databases as defined in **TP**.
FMT_MTD.1 User Attributes	User security attributes are protected as required by the user identification and authentication data management **IA** and during the creation of new users in **SM**. User attributes are stored in TSF databases described in **TP**.
FMT_MTD.1 Authen. Data	Initialization of authentication data is restricted to administrative users during the creation of new users in **SM**. Authentication data (in encrypted form) and attributes are stored in TSF databases described in **TP**. Users are allowed to change their own authentication data within the limits defined by an administrative user. This is described in **SM**
FMT_REV.1 User Attributes	The revocation of user security attributes as required in FMT_REV.1 is realized by the user management functions of **SM**.
FMT_REV.1 Obj. Attributes	Revocation of object security attributes is realized by the access control configuration and management function **SM**.
FMT_SMF.1	Management of security functions is addressed in the following security functions: Object security attributes management: **DA** (File system objects and IPC objects). In addition the following management functions are defined: User attribute management**: SM** Authentication management: **SM** and **IA** In addition most of the management functions use the TSF databases (**TP**) to store management configurations.
FMT_SMR.1	The required roles are maintained within the security management of the roles in function **SM**.
FPT_RVM.1	The TSF invocation guarantee functionality **TP** ensure that TSP enforcement functions are always invoked before functions in the TSC are allowed to proceed.
FPT_SEP.1	The required domain separation for the TSF is realized by the kernel functionality itself, the kernel modules and trusted processes as described in **TP**, the discretionary access control mechanism described in **DA** and the internal TOE protection mechanisms described in **TP**.

This table shows, how the security functions work together to satisfy the security functional requirements.

Access control is defined by a discretionary access control policy in FDP_ACC.1 and FDP_ACF.1. For SLES there are two different types of objects with some differences in policies depending on the object type. All the dependencies on the management aspects have been resolved. The management of the two object types differ only slightly, where those differences are explained in FMT_MSA.1 and FMT_REV.1.

Object reuse is a useful requirement to prohibit unwanted access to information via resources that have not been prepared for reuse. Since the TOE supports access control, object resuse makes sense. This is addressed in FDP_RIP.2.

Identification and authentication is handled by FIA_ATD.1, FIA_SOS.1 FIA_UAU.2, FIA_UAU.7 FIA_UID.2 and FIA_USB.1 in a fairly conventional way. FIA_USB describes the way the effective user ID and group ID can be changed.

In the management section the requirements for the management User Attributes and Authentication Data has been separated in this Security Target. Since they are clearly separated, they are not contradicting each other.

Revocation for user attributes is described separately from revocation of object attributes in two instantiations of FMT_REV.1. This makes sense, since revocation is handled differently. FMT_SMF.1 has been inlcuded because of AIS 32 Final Interpretation 065 and covers the different management aspects addressed in detail in FMT_MSA.1 and the instantiations of FMT_MTD.1.

The TOE supports only two different roles as expressed by FMT_SMR.1. No additional role is required by any other SFR, so the role model is consistent with the other requirements.

FPT_RVM.1 is required to ensure that the security functions can not be bypassed. In addition FPT_SEP.1 ensures that untrusted programs can not tamper with the TSF and cause them to operate in contradiction to the security policy of the TOE. FPT_AMT.1, FPT_RVM.1 and FPT_SEP.1 are therefore mutually supportive requirements to enable a sufficient self-protection of the TSF.

As a summary this shows that the security functional requirements are not contradicting each other and are mutually

supportive.

8.3.2 Assurance Measures Justification

The TOE summary specification in section 6.4 includes a justification that each TOE security assurance requirement is met by appropriate assurance measures.

8.3.3 Strength of function

The password mechanism used for authentication is the only mechanism in the TSF that is implemented by a permutational or probabilistic mechanism. For the password based authentication mechanism of the security function IA.1, a minimum strength of SOF-basic is claimed. This is done in accordance with the SOF claim for the related security functional requirement FIA_SOS.1. There is no other mechanism in this Security Target where a strength of function claim is required.

This claims is consistent with the security objctive O.AUTHORIZATION and the statement in section 3.2 which says that the TOE should „protect against threats of inadvertent or casual attempts to breach the system security". A highly skilled and well funded attacker is explicitely excluded from the threat scenario described in section 3.2. Therefore a strength of SOF-basic is consistent with the description of the TOE environment.

8.4 PP Claims Rationale

No compliance with any existing Protection Profile is claimed.

9 Abbreviations

ACL	Access Cotrol List
AIX	Advanced Interactive Executive
ANSI	American National Standards Institute
CAPP	Controlled Access Protection Profile
CC	Common Criteria
CD	Compact Disc
CPU	Central Processing Unit
DAC	Discretionary Access Control
DVD	Digital Versatile Disc
FPR	Floating Point Register
FSO	File System Object
FTP	File Transfer Protocol
GPR	General Purpose Register
ID	Identifier
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
IPC	Inter-Process Communication
LAN	Local Area Network
ISO	International Standards Organization
MD5	Message Digest 5
PAM	Pluggable Authentication Module
PCMCIA	Personal Computer Memory Card International Association
PDF	Portable Data Format
PP	Protection Profile
SLES	SuSE Linux Enterprise Server
SSH	Secure Shell
ST	Security Target
TCP	Transmission Control Protocol
TOE	Target of Evaluation
TSF	TOE Security Functions
UDP	User Datagram Protocol
VFS	Virtual File System
VMM	Virtual Memory Manager