SUSE Linux Enterprise Server V 9
Security Target
for CAPP Compliance

Version: 3.10

Last Update: 2005-01-17

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

Document History

Version	Date	Changes	Summary	Author
0.1	2003-02-20	no	Initial Version	Doc Shankar, IBM Helmut Kurth, atsec
0.2	2003-02-24	a lot	Modifications is almost all chapters to address comments Audit was removed completely Integrated comments from Doc	Doc Shankar, IBM Helmut Kurth, atsec
0.3	2003-02-27	a lot	Modifications due to comments from Roman, Andi, Doc, Ken, Daniel and Emily	Helmut Kurth, atsec
1.0	2003-03-01	few	Integrated last comments	Helmut Kurth, atsec
1.1	2003-04-10	quite some	Integrated comments, went from telnet and r-commands to ssh, clarified some issues around access control, detailed password policy, included final package list and list of TSF databases	Helmut Kurth, atsec
1.2	2003-05-09	few	Integrated some comments from Dan, Gerald, Klaus, Roman, Andreas Modified the description of the handling of kernel modules to reflect the modifications in the SLES 8 kernel.	Helmut Kurth, atsec
1.3	2003-05-09	few	Modified description of permission bits (no change in functionality, just to avoid potential misinterpretation). Corrected some spelling errors, updated list of abbreviations.	Helmut Kurth, atsec
1.4	2003-05-15	minor	section 6.2.3.2.10 has been modified such that bind now only requires only write access, not read and write access to the socket file. This has been identified during testing and discussed with SuSE. This is not seen as an error since it does not contradict the man page and does not have any other impact on security functions. Further some typos have been corrected and the terminology has been harmonized. Added the SLES Security Guide to the references.	Helmut Kurth, atsec
1.5	2003-05-19	editorial	Some editorial changes for better readability and understanding. No change of actual content.	Helmut Kurth, atsec
2.0	2003-06-20	major	First draft to include the requirements for CAPP compliance and additional functions. Transformed from WordPro to Word (going from one hell to another)	Helmut Kurth, atsec
2.1	2003-06-30	major	Finished the ST with CAPP compliance, integrated comments from various people.	Helmut Kurth, atsec
2.2	2003-07-09	major	Integrated further details for audit. Integrated details on ssh and ssl. Updated chapter 8.	Helmut Kurth, atsec
2.3	2003-07-14	minor	Fixed a few typos and minor errors	Helmut Kurth, atsec
2.4	2003-08-01	minor	Fixed a few items identified during the evaluation	Helmut Kurth, atsec
2.5	2003-10-07	minor	Adressing comments from the developers and the certifiers. Adapting some aspects of the audit function to the real implementation. Including some clarifications on how to address the cryptographic functions	Helmut Kurth, atsec
2.6	2003-10-20	minor	Added a copyright notice changed the audit event table in FAU_GEN.1.1 to reflect the event names used in the audit	Helmut Kurth, atsec

Version	Date	Changes	Summary	Author
			implementation, added eServer 325 (Opteron) as additional hardware platform, removed AES as cryptographic algorithm.	
2.7	2003-11-11	minor	modified “part 2 conformant” to “part 2 extended” (caused by the (in)famous “Note 1” from CAPP. Mentioned that there may be more than 2 file for audit in bin mode. Fixed some formatting problems. Removed the RC4 with MD5 cipher suite from requirements in chapter 5 to be consistent with chapter 6. Finalized list of packages. Removed references to /var/log/faillog (does not work as described and failed login attempts are recorded by the audit subsystem) Added an application note to FAU_STG.3 describing that the situation defined there can not occur in bin mode. Also extended the explanation how audit works in chapter 6. Removed the Triple-DES cipher suite from the SSL function. The reason is an interoperability problem when used with the SSL implementation in Opera and Internet Explorer.	Helmut Kurth, atsec
3.0	2004-02-05	reflecting EAL4	Modified the EAL3 Security Target such that it can be used as a starting point for the EAL4 evaluation.	Helmut Kurth, atsec
3.1	2004-02-13	minor editorial	addressing comments from IBM and SUSE	Helmut Kurth, atsec
3.2	2004-03-05	minor editorial	addressing comments from the evaluator	Helmut Kurth, atsec
3.3	2004-03-08	minor editorial	addressing two additional comments from the evaluator	Helmut Kurth, atsec
3.4	2004-07-07	minor editorial	Updating the OS/400 LPAR Version number Updating the (interim) list of packages	Helmut Kurth, atsec
3.5	2004-08-20	minor	Updating table in TP.5 with the latest changes	Helmut Kurth, atsec
3.6	2004-11-18	minor	Last updates with respect to list of packages, SSL cipher suites, hardware platforms. Integrating comments from evaluators and the certification body.	Helmut Kurth, atsec
3.7	2004-12-01	minor	Adjustment to descriptions of the file system supported.	Helmut Kurth, atsec
3.8	2004-12-15	minor	Adressing an inconsistency with the number of file systems supported. Added an assumption on the seeding of the DRNG. Updating the version of the Evaluated Configuration Guide.	Helmut Kurth, atsec
3.9	2005-01-07	minor	After discussion with BSI the assumption on the seeding of the DRNG was removed.	Helmut Kurth, atsec
3.10	2005-01-17	minor	Updated version of kernel package	Helmut Kurth, atsec

Table of Content

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]	Common Criteria EAL4+ Evaluated Configuration Guide for SUSE LINUX Enterprise Server on IBM Hardware, v1.20, January 17, 2005
[SSLv3]	Alain O. Freier, Philip Karlton, Paul C. Kocher: The SSL Protocol, Version 3; IETF Memo, Internet Draft, November 1996
[SSHv2]	Internet Draft: SSH Transport Layer Protocol (draft-ietf-secsh-transport-15.txt)
[RFC3268]	P. Chown: Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security (TLS), IETF, June 2002
[ST-AIX-CC]	AIX 5.2 Security Target
[TARGET]	SUSE Linux Enterprise Server 9 Security Target for CAPP compliance (this document)
[X.509]	ITU-T RECOMMENDATION X.509 | ISO/IEC 9594-8: INFORMATION TECHNOLOGY - OPEN SYSTEMS INTERCONNECTION - THE DIRECTORY: PUBLIC-KEY AND ATTRIBUTE CERTIFICATE FRAMEWORKS

1 Introduction

This is version 3.10 of the Security Target document for the evaluation of SUSE Linux Enterprise Server Version 9 and the certification-sles-ibm-eal4.rpm package. This Security Target has been derived from the Security Target used for the previous evaluation of "SUSE Linux Enterprise Server 8 with Service Pack 3 RC4 and the certification-sles-eal3" package at the EAL3+ level. The major changes to this Security Target are:

- an evaluation assurance level of EAL4 augmented by ALC_FLR.3 (was EAL3 augmented by ALC_FLR.2).
- changing from SLES 8 SP3 which was based on the 2.4.21 kernel to SLES 9 which is based on the 2.6.5 kernel (which results in the inclusion of the new sysfs file system).
- some functions of the TOE related to cryptography are now also subject to the strength of function analysis. Note that cryptographic algorithms themselves are still not rated for their cryptographic strength. People interested into this subject should read the various papers discussing the cryptographic algorithms implemented within the TOE.

1.1 ST Identification

Title: SUSE Linux Enterprise Server 9 Security Target with the certification-sles-ibm-eal4 package, Version 3.10

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 9) operating system product, and is conformant to the Common Criteria for Information Technology Security Evaluation [CC] with extensions as defined in the Controlled Access Protection Profile [CAPP].

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 9) with the certification-sles-ibm-eal4.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 all 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. This Security Target therefore claims full compliance with the requirements of this Protection Profile and also includes additional functional and assurance packages beyond those required by CAPP.

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. Communication links can be protected against loss of confidentiality and integrity by security functions of the TOE based on cryptographic protection mechanisms.

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 extended* and *Part 3 conformant*, with a claimed Evaluation Assurance Level of EAL4 augmented by ALC_FLR.3.

The extensions to part 2 of the Common Criteria are those introduced by the Controlled Access Protection Profile [CAPP].

1.4 Strength of Function

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

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 9", 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 require 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 assigned 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.

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 9 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 9 (SLES) with the certification-sles-ibm-eal4.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.

The SLES evaluation covers a potentially distributed, but closed network of IBM xSeries, pSeries, zSeries, iSeries and eServer 325 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. The basic tools required for the secure configuration and management of the TOE are included as part of the TSF in this evaluation. Other tools exist that can be used for configuration and management functions have not been part of this evaluation.

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

The TOE includes installation from CDROM (if available) and/or from a local hard disk partition.

The TOE includes standard networking applications, such as ftp, stunnel 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. The Evaluated Configuration Guide [SECGUIDE] provides guidance how to set up an HTTP server on the TOE in a secure way.

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 named 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 a 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)
    - pam_passwdqc.so (for additional password checking)

  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.
- A dedicated auditing subsystem. This auditing subsystem allows for the auditing of security critical events and provides tools for the administrative user to configure the audit subsystem and evaluate the audit records.
- Basic hardware check functions. They allow an administrative user to check on demand if the basic security functions of the hardware the TOE relies upon are provided correctly.

2.2 Summary of Security Features

The primary security features of the product are:

- Identification and Authentication
- Audit
- Discretionary Access Control
- Object reuse functionality
- Security Management
- Secure Communication
- 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 medium password strength and limit the use of the su command and restrict root login to specific terminals are also included.

2.2.2 Audit

The TOE provides an audit capability that allows generating audit records for security critical events. The administrative user can select, which events are audited, for which users auditing is active and also has tools available to extract audit records from the audit trail using defined selection criteria. A list of events that can be audited is defined in chapter 5 and 6.

The TOE provides tools for the administrative user that allow him to extract specific types of audit events, audit events for specific users, audit events related to specific file system objects or audit events within a specific time frame from the overall audit records collected by the TOE. Those tools allow an administrative user to save or print the selected audit records in human readable format.

The audit function informs the system administrator via a syslog message when the capacity of the audit trail exceeds a configurable limit. The audit function also ensures that no audit records get lost due to exhaustion of the internal audit buffers. Processes that try to create an audit record while the internal audit buffers are full will be halted until the required resources are available again.

2.2.3 Discretionary Access Control

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

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

2.2.4 Object Reuse

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

2.2.5 Security Management

The management of the security critical parameters of the TOE 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.

2.2.6 Secure Communication

The TOE supports secure communication with other systems via the SSH v2 and SSL v3 protocol. Communication via those protocols is protected against unauthorized disclosure and modification via cryptographic mechanisms. The TOE also allows for secure authentication of the communicating parties using the SSL v3 protocol with client and server authentication. This allows establishing a secure communication channel between different machines running the TOE even over an insecure network. The SSL v3 protocol can be used to tunnel otherwise unprotected protocols in a way that allows an application to secure its TCP based communication with other servers (provided the protocol uses a single TCP port). To do this stunnel is used as a trusted application started by xinetd.

2.2.7 TSF Protection

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

Non-kernel TSF software and data are protected by DAC and process isolation mechanisms. In 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 hardware components that are protected from direct access by user programs. A user process may execute unprivilged instructions and read or write to memory and processor regsister within the bounds defined by the kernel for the user process without those types of access being mediated by the kernel. All other types of access to hardware resources by user processes can only be performed by requests (in the form of system calls) to the kernel.

The TOE provides a tool that allows an administrative user to check the correct operation of the underlying hardware. This tool performs tests to check the system memory, the memory protection features of the underlying processor and the correct separation between user and supervisor state.

2.3 Software

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

- SUSE Linux Enterprise Server Version 9 with the certification-sles-ibm-eal4.rpm package

The TOE and its documentation is supplied on CD-ROM except for the certification-sles-ibm-eal4.rpm package which needs to be downloaded from the SUSE web site. This package contains the Evaluated Configuration Guide [SECGUIDE], 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.

The list contains the packages with their version numbers. The list starts with the packages that are available (with the same version) on all platforms followed by the list of platform specific packages or packages where the version is different for different platforms. Those packages are listed for each of the evaluated platforms.

All platforms:

aaa_base	9-29.13
aaa_skel	2004.6.8-0.2
acl	2.2.21-54.1
amtu	0.1-1.6
ash	0.4.18-56.1
at	3.1.8-898.1
attr	2.4.12-56.1
autoyast2-installation	2.9.45-0.2
bash	2.05b-305.6
bc	1.06-744.1
bzip2	1.0.2-346.1
certification-sles-ibm-eal4	1.0
core-release	9-6.3
coreutils	5.2.1-23.5
cpio	2.5-324.1
cracklib	2.7-1006.1
cron	3.0.1-920.1
curl	7.11.0-39.1
cyrus-sasl	2.1.18-33.1
db	4.2.52-86.3
device-mapper	1.00.09-17.5
devs	9-16.8
dialog	0.9b-188.1
diffutils	2.8.4-75.1
dosfstools	2.10-90.1
e2fsprogs	1.34-115.1
ed	0.2-864.1
evms	2.3.3-0.15
file	4.07-48.5
filesystem	9-29.4
fillup	1.42-98.1
findutils	4.1.7-860.1
gawk	3.1.3-210.1
gdbm	1.8.3-228.1
glibc	2.3.3-98.31
glibc-locale	2.3.3-98.31
gpg	1.2.4-68.1
gpm	1.20.1-301.1
grep	2.5.1-427.1
groff	1.17.2-881.1
gzip	1.3.5-136.1
hdparm	5.5-41.3
heimdal-lib	0.6.1rc3-55.3

hfsutils	3.2.6-1038.1
hotplug	0.44-32.21
howtoenh	2004.4.4-0.4
hwinfo	8.61-0.3
info	4.6-61.1
insserv	1.00.2-85.1
iproute2	2.4.7-866.5
iputils	ss021109-147.1
kbd	1.12-26.1
ksymoops	2.4.9-135.1
laus	0.2-14.17
ldapcpplib	0.0.3-21.3
less	382-34.8
libacl	2.2.21-54.1
libattr	2.4.12-56.1
libgcc	3.3.3-43.24
libselinux	1.8-16.1
libstdc++	3.3.3-43.24
libxcrypt	2.1.90-61.3
libxml2	2.6.7-28.1
liby2util	2.9.25-0.2
logrotate	3.7-31.1
lprng	3.8.25-37.1
lsof	4.70-30.1
lukemftp	1.5-578.1
m4	1.4o-622.1
mailx	10.6-65.1
man	2.4.1-214.1
man-pages	1.67-1.6
mingetty	0.9.6s-73.1
mkinitrd	1.0-199.53
mktemp	1.5-729.1
module-init-tools	3.0_pre10-37.16
ncurses	5.4-61.4
net-tools	1.60-543.6
netcat	1.10-864.1
netcfg	9-17.1
openldap2-client	2.2.6-37.19
openslp	1.1.5-73.9
openssh	3.8p1-37.9
openssl	0.9.7d-15.13
pam-laus	0.77-4.3
pam-modules	9-18.5
parted	1.6.6-138.3
pciutils	2.1.11-192.8
pcre	4.4-109.1
perl	5.8.3-32.1
perl-Config-Crontab	1.03-46.1
permissions	2004.7.30-0.2
popt	1.7-176.7
postfix	2.1.1-1.4
procps	3.2.1-5.3
psmisc	21.4-39.1
pwdutils	2.6.4-2.16
readline	4.3-306.5
release-notes	9.1-8.40
resmgr	0.9.8-47.3
rpm	4.1.1-177.6
scsi	1.7_2.34_1.06_0.11-9.9
sed	4.0.9-31.1
sitar	0.8.11-17.1
sles-admin_en	9.1.0.5-0.16
sles-release	9-82.11
star	1.5a38-26.1
stunnel	4.05-20.1
submount	0.9-33.6
suse-build-key	1.0-662.4
sysconfig	0.31.0-15.29

```
        syslogd                      1.4.1-519.3
        sysvinit                     2.85-21.3
        tar                          1.13.25-325.3
        tcpd                         7.6-710.1
        telnet                       1.1-38.3
        terminfo                     5.4-61.4
        texinfo                      4.6-61.1
        timezone                     2.3.3-98.31
        udev                         021-36.32
        usbutils                     0.11-211.1
        utempter                     0.5.2-385.4
        util-linux                   2.12-72.20
        vim                          6.2-235.1
        vsftpd                       1.2.1-69.3
        w3m                          0.4.1_m17n_20030308-201.1
        wget                         1.9.1-45.3
        xinetd                       2.3.13-39.3
        yast2                        2.9.75-0.2
        yast2-bootloader             2.9.34-0.3
        yast2-core                   2.9.94-1.2
        yast2-country                2.9.24-0.2
        yast2-inetd                  2.9.12-21.1
        yast2-installation           2.9.89-0.2
        yast2-ldap                   2.9.15-1.2
        yast2-ldap-client            2.9.23-0.2
        yast2-mail-aliases           2.9.16-0.2
        yast2-mouse                  2.9.11-4.1
        yast2-ncurses                2.9.26-0.2
        yast2-network                2.9.59-0.2
        yast2-online-update          2.9.12-0.3
        yast2-packagemanager         2.9.51-1.3
        yast2-packager               2.9.51-0.2
        yast2-pam                    2.9.13-0.3
        yast2-perl-bindings          2.9.34-1.2
        yast2-runlevel               2.9.15-0.2
        yast2-security               2.9.14-18.1
        yast2-storage                2.9.58-0.3
        yast2-sysconfig              2.9.15-0.2
        yast2-theme-SuSELinux        2.9.13-0.4
        yast2-trans-en_US            2.9.7-1.2
        yast2-transfer               2.9.3-0.2
        yast2-update                 2.9.27-0.2
        yast2-users                  2.9.39-0.3
        yast2-x11                    2.9.11-0.2
        yast2-xml                    2.9.8-19.1
        zlib                         1.2.1-70.6

Only on i386:
        grub                         0.94-45.3
        isapnp                       1.26-489.1
        kernel-default               2.6.5-7.111-30
        kernel-smp                   2.6.5-7.111-30
        lilo                         22.3.4-508.1

Only on x86_64:
        bzip2-32bit                  9-200407011229
        cracklib-32bit               9-200407011229
        cyrus-sasl-32bit             9-200407011229
        db-32bit                     9-200407011229
        e2fsprogs-32bit              9-200407011229
        file-32bit                   9-200407011229
        gdbm-32bit                   9-200407011229
        glibc-32bit                  9-200407011233
        glibc-locale-32bit           9-200407011229
        grub                         0.94-45.3
        heimdal-lib-32bit            9-200407011229
        irqbalance                   0.09-37.1
        isapnp                       1.26-489.1
        kernel-default              2.6.5-7.111.30
        kernel-smp                  2.6.5-7.111.30
        laus-32bit                  9-200407011229
        libacl-32bit                9-200407011229
        libattr-32bit               9-200407011229
        libselinux-32bit            9-200407011229
        libxcrypt-32bit             9-200407011229
        libxml2-32bit               9-200407011229
        lilo                        22.3.4-508.1
        ncurses-32bit               9-200407011229
        openldap2-client-32bit      9-200407011229
        openssl-32bit               9-200407011229
        pcre-32bit                  9-200407011229
        perl-32bit                  9-200407011229
        popt-32bit                  9-200407011229
        readline-32bit              9-200407011229
        resmgr-32bit                9-200407011229
        utempter-32bit              9-200407011229
        zlib-32bit                  9-200407011229

Only on s390x:
        bzip2-32bit                 9-200407011411
        cracklib-32bit              9-200407011411
        cyrus-sasl-32bit            9-200407011411
        db-32bit                    9-200407011411
        e2fsprogs-32bit             9-200407011411
        file-32bit                  9-200407011411
        gdbm-32bit                  9-200407011411
        glibc-32bit                 9-200407011411
        glibc-locale-32bit          9-200407011411
        heimdal-lib-32bit           9-200407011411
        kernel-s390x                2.6.5-7.111.30
        laus-32bit                  9-200407011411
        libacl-32bit                9-200407011411
        libattr-32bit               9-200407011411
        libgcc-32bit                9-200407011411
        libselinux-32bit            9-200407011411
        libxcrypt-32bit             9-200407011411
        libxml2-32bit               9-200407011411
        ncurses-32bit               9-200407011411
        openldap2-client-32bit      9-200407011411
        openssl-32bit               9-200407011411
        pcre-32bit                  9-200407011411
        perl-32bit                  9-200407011411
        popt-32bit                  9-200407011411
        readline-32bit              9-200407011411
        resmgr-32bit                9-200407011411
        s390-tools                  1.3.1-0.3
        utempter-32bit              9-200407011411
        zlib-32bit                  9-200408261522

Only on iSeries:
        baselibs-64bit              9-200407011606
        bzip2-64bit                 9-200407011606
        cracklib-64bit              9-200407011606
        cyrus-sasl-64bit            9-200407011606
        db-64bit                    9-200407011606
        e2fsprogs-64bit             9-200407011606
        file-64bit                  9-200407011606
        gdbm-64bit                  9-200407011606
        glibc-64bit                 9-200407011606
        glibc-locale-64bit          9-200407011606
        heimdal-lib-64bit           9-200407011606
        kernel-pseries64            2.6.5-7.111.30
        laus-64bit                  9-200407011606
        libacl-64bit                9-200407011606
        libattr-64bit               9-200407011606
        libgcc-64bit                9-200407011606
        libselinux-64bit             9-200407011606
        libstdc++-64bit              9-200407011606
        libxcrypt-64bit              9-200407011606
        libxml2-64bit                9-200407011606
        lilo                         0.0.15-22.17
        ncurses-64bit                9-200407011606
        openldap2-client-64bit       9-200407011606
        openssl-64bit                9-200407011606
        pcre-64bit                   9-200407011606
        pdisk                        0.8a-445.1
        perl-64bit                   9-200407011606
        popt-64bit                   9-200407011606
        ppc64-utils                  0.9-1.1
        readline-64bit               9-200407011606
        resmgr-64bit                 9-200407011606
        utempter-64bit               9-200407011606
        zlib-64bit                   9-200407011606

Only on pSeries:
        bzip2-64bit                  9-200407011606
        cracklib-64bit               9-200407011606
        cyrus-sasl-64bit             9-200407011606
        db-64bit                     9-200407011606
        e2fsprogs-64bit              9-200407011606
        file-64bit                   9-200407011606
        gdbm-64bit                   9-200407011606
        glibc-64bit                  9-200407011606
        heimdal-lib-64bit            9-200407011606
        kernel-pseries64             2.6.5-7.111.30
        laus-64bit                   9-200407011606
        libacl-64bit                 9-200407011606
        libattr-64bit                9-200407011606
        libgcc-64bit                 9-200407011606
        libselinux-64bit             9-200407011606
        libstdc++-64bit              9-200407011606
        libxcrypt-64bit              9-200407011606
        libxml2-64bit                9-200407011606
        lilo                         0.0.15-22.17
        ncurses-64bit                9-200407011606
        openldap2-client-64bit       9-200407011606
        openssl-64bit                9-200407011606
        pcre-64bit                   9-200407011606
        pdisk                        0.8a-445.1
        perl-64bit                   9-200407011606
        popt-64bit                   9-200407011606
        ppc64-utils                  0.9-1.1
        readline-64bit               9-200407011606
        resmgr-64bit                 9-200407011606
        utempter-64bit               9-200407011606
        zlib-64bit                   9-200407011606
```

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 Evaluated Configuration Guide [SECGUIDE] and installed accordingly.
- SLES supports the use of IPv4 and IPv6, only IPv4 is included within the TOE.
- 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 system and afforded the same physical protection as the server.

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 synchronized for the different servers this synchronization 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. Additional restrictions apply for specific objects in this file system.
- The sysfs filesystem (sysfs) used to export and handle non-process related kernel information like driver specific information.
- The temporary filesystem (tmpfs) used as a temporary RAM based file system.
- The pseudoterminal device file system (devpts) used to provide pseudo terminal support.

2.4.2 Technical Environment for Use

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

The TOE is running on the following hardware platforms:

- IBM xSeries - model x335 machine type 8676
- IBM zSeries - model z990
  (the TOE executes in a VM 5.1 Logical Partition)
- IBM iSeries - model 520 machine type (9406) (OS/400 V5R3 LPAR)
- IBM pSeries - model 520 machine type 9111 with pSeries LPAR SF220_049
- IBM eServer - model 325 (based on the AMD64 (Opteron) processor), machine type 8835

The following peripherals can be used 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). **Note:** Serial devices are not supported on iSeries and zSeries.
- 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

Note: the peripherals are physical peripherals for the xSeries, pSeries and eServer models. In the case of zSeries and iSeries the TOE is executing within a logical partition and the peripherals used may be virtualized. The logical partitioning software is part of the “abstract machine” and therefore part of the TOE environment. The Evaluated Configuration Guide [SECGUIDE] provides the required guidance on how to set up and configure the logical partitioning software and how to define the logical peripheral devices such that the TOE operates securely in the logical partitioning environment on the zSeries and iSeries platforms.

Note: Excluding hot pluggable devices connected via USB does not exclude all USB devices. USB keyboards and mice may be attached provided they are connected before booting the operating system.

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 organizational 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 security vulnerabilities that might be exploited in the intended environment for the TOE with medium level of expertise and effort. The TOE in accordance with the strength of function claimed protects against straightforward or intentional breach of TOE security by attackers possessing a low attack potential.

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.

T.COMPROT An attacker (possibly, but not necessarily, an unauthorized user of the TOE) may intercept a communication link between the TOE and another trusted IT product to intercept or modify information transferred between the TOE and the other trusted IT product (which may be another instantiation of the TOE) using defined protocols (SSH or SSL) in a way that can not be detected by the TOE or the other trusted IT product.

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 An attacker with legitimate physical access to the hardware of the TOE (examples are maintenance personnel or legitimate users) or environmental conditions may cause a hardware malfunction with the effect that a user (normal or administrative) is losing stored data due to this hardware malfunction. An attacker may cause such a hardware malfunction either by having physical access to the hardware the TOE is running on or by executing software that capable of causing hardware malfunction. Note that such a hardware malfunction may be caused accidently without malicious intent by persons having physical access to the TOE.

TE.COR_FILE An attacker (possibly, but not necessarily, an unauthorized user of the TOE) or environmental conditions like a hardware malfunction may intentionally or accidentally modify or corrupt security enforcing or relevant files of the TOE without an administrative user being able to detect this. An attacker may corrupt such files either by having physical access to the hardware the TOE is running on, by booting other software than the TOE in its evaluated configuration or by modifying or corrupting files on backup media. Note that such a corruption may be caused accidently without malicious intent by persons having legitimate access to media where such data is stored.

TE.HW_SEP An attacker (possibly, but not necessarily, an unauthorized user of the TOE) with legitimate physical access to the hardware the TOE is running on or environmental conditions may cause the underlying hardware functions of the hardware the TOE is running on to not provide sufficient capabilities to support the self-protection of the TSF from unauthorized programs. Note that this also covers persons with legitimate access to the TOE hardware causing such a problem accidently without malicious intent.

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.

P.ACCOUNTABILITY

The users of the system shall be held accountable for their actions within the system.

3.4 Assumptions

This section indicates the minimum physical and procedural measures required to maintain security of the SLES 9 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 exercising 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 not using the secured protocols SSH v2 or SSL v3 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.AUDITING The TSF must record the security relevant actions of users of the TOE. The TSF must present this information to authorized administrators.

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 administrative 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 organizational policies are enforced in the target environment i.e. the integrity of the TSF is protected.

O.COMPROT The TSF must be designed and implemented in a manner that allows for establishing a trusted channel between the TOE and another trusted IT product that protect the user data transferred over this channel from disclosure and undetected modification.

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.

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 unless one of the secure protocols provided by the TOE is used for the communication with another trusted entity.

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 Administrative 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. The TOE provides some security functions that can be used to protect communication links, but the TOE does not enforce that those functions are used for all communication links. Communication links not protected by the functions provided as part of the TOE or communication links that need protection against interruption of communication have to be protected by security measures in the TOE environment..)

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 (when using communication links not protected by the use of the SSL or SSH protocols. Note that interruption of communication is not prevented by the use of those protocols and if protection against interruption of communication is required, adequate protection in the TOE environment has to be established for all communication links!).

5 Security Requirements

5.1 TOE Security Functional Requirements

Most of the following security functional requirements are taken from the "Controlled Access Protection Profile", Version 1.d [CAPP]. One requirement (FMT_SMF.1) has been added due to AIS 32 / Final Interpretation 065 that has been published after the CAPP had been issued. Other requirements (FCS_CKM.1, FCS_CKM.2, FCS_COP.1, FDP_UCT.1, FDP_UIT.1, FMT_MSA.2 and FTP_ITC.1) represent TOE specific extensions to the requirements defined by [CAPP].

For easier comparison with [CAPP] the Application Notes and the Rationale presented in [CAPP] for each security functional requirement have been repeated in this Security Target. They have been marked as "Application Note (CAPP)" and "Rationale (CAPP)" to remind the reader where this text comes from. The Application Notes of [CAPP] mainly provide some guidance and requirements for the author of a Security Target. The reader can than easily see how those requirements have been addressed within this Security Target.

[CAPP] has already performed some instantiations and even some refinements of the security functional requirements as defined in the Common Criteria. Those instantiations and refinements are marked in **bold** within each of the requirements. In addition this Security Target has instantiated and refined the requirements as stated in [CAPP]. Those instantiations and refinements that are specific for this Security Target are marked in ***bold, italic and blue***.

Security Functional requirements in addition to those taken from [CAPP] are shown in green with TOE specific instantiations marked in ***green, bold and italic***.

5.1.1 Security Audit (FAU)

5.1.1.1 Audit Data Generation (FAU_GEN.1)

FAU_GEN.1.1 The TSF shall be able to generate an audit record of the auditable events **listed in column "Event" of Table 5-1 (Auditable Events). This includes all auditable events for the basic level of audit, except FIA_UID.2's user identity during failures.**

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

a) Date and time of the event, type of event, subject identity, and the outcome (success or failure) of the event;

b) The additional information specified in the "Details" column of Table 5-1 (Auditable Events).

Application Note (CAPP) : For some situations it is possible that some events cannot be automatically generated. This is usually due to the audit functions not being operational at the time these events occur. Such events need to be documented in the Administrative Guidance, along with recommendation on how manual auditing should be established to cover these events.

Rationale (CAPP): This component supports O.AUDITING by specifying the detailed, security relevant events and data that the audit mechanism must be capable of generating and recording. The "basic" level of auditing was selected as best representing the "mainstream" of contemporary audit practices used in the target environments.

Table 5-1: Auditable Events

Component	Event	Details (Event Names)
FAU_GEN.1	Start-up and shutdown of the audit functions.	Events AUDIT_start, AUDIT_stop (from auditd)
FAU_GEN.2	None	
FAU_SAR.1	Reading of information from the audit records.	syscall *open* (on the audit log files)

Component	**Event**	**Details (Event Names)**
FAU_SAR.2	Unsuccessful attempts to read information from the audit records.	like FAU_SAR.1, but with negative results
FAU_SAR.3	None	
FAU_SEL.1	All modifications to the audit configuration that occur while the audit collection functions are operating.	Events AUDCONF_reload (generated by `auditd`); syscalls *open*, *link*, *unlink*, *rename*, *truncate* (write access to configuration files)
FAU_STG.1	None	
FAU_STG.3	Actions taken due to exceeding of a threshold.	Event AUDIT_disklow (generated by `auditd`); execution of administrator-specified alert program
FAU_STG.4	Actions taken due to the audit storage failure.	Event AUDIT_diskfull (generated by `auditd`); execution of administrator-specified alert program; all audited actions are blocked (process sleeps until space becomes available)
FCS_CKM.1	None	
FCS_CKM.1	None	
FCS_CKM.1	None	
FCS_CKM.2	None	
FCS_CKM.2	None	
FCS_CKM.2	None	
FCS_CKM.2	None	
FCS_COP.1	None	
FCS_COP.1	None	
FCS_COP.1	None	
FDP_ACC.1	None	
FDP_ACF.1	All requests to perform an operation on an object covered by the SFP.	syscalls *chmod*, *chown*, *setxattr*, *removexattr, link*, *symlink, mknod*, *open*, *rename*, *truncate*, *unlink*, *rmdir*, *mount*, *umount*, *msgctl*, *msgget*, *semget*, *semctl*, *semop*, *shmget*, *shmctl*; details include identity of object
FDP_RIP.2	None	
Note 1	None	
FDP_UCT.1	None	
FDP_UIT.1	None	
FIA_ATD.1	None	
FIA_SOS.1	Rejection or acceptance by the TSF of any tested secret.	Events AUTH_success, AUTH_failure (from PAM framework, subtypes ”authentication” and “bad ident”); details include origin of attempt (terminal or IP address as applicable)

Component	Event	Details (Event Names)
FIA_UAU.2	All use of the authentication mechanism.	Events AUTH_success, AUTH_failure (from PAM framework, “authentication” subtype)
FIA_UAU.7	None	
FIA_UID.2	All use of the user identification mechanism, including the identity provided during successful attempts.	Events AUTH_success, AUTH_failure (from PAM framework, “authentication” subtype)
FIA_USB.1	Success and failure of binding user security attributes to a subject (e.g. success and failure to create a subject).	Success: LOGIN audit record (from `pam_laus.so` module or `aurun`); syscall *vfork* Failure: Event AUTH_failure (from PAM framework, “authentication” subtype
FMT_MSA.1	All modifications of the values of security attributes.	syscalls *chmod*, *chown*, *setxattr*, *msgctl*, *semctl*, *shmctl*
FMT_MSA.2	None	
FMT_MSA.3	Modifications of the default setting of permissive or restrictive rules. All modifications of the initial value of security attributes.	syscalls *umask*, *open*
FMT_MTD.1 Audit Trail	All modifications to the values of TSF data.	syscalls *open*, *rename*, *link*, *unlink*, *truncate* (of audit log files)
FMT_MTD.1 Audit Events	All modifications to the values of TSF data.	syscalls *open*, *link*, *rename*, *truncate*, *unlink* (of audit config files); event AUDCONF_reload
FMT_MTD.1 User Attributes	All modifications to the values of TSF data.	``gpasswd'' audit text messages (from shadow suite), details include new value of of the TSF data
FMT_MTD.1 Authentication Data	All modifications to the values of TSF data.	“gpasswd” audit text messages (from shadow suite); attempts to bypass trusted programs detected through audited syscalls *open*, *rename*, *truncate*, *unlink*
FMT_REV.1	All attempts to revoke security attributes.	Event: ``gpasswd'' audit text messages (from shadow suite); attempts to bypass trusted programs detected through audited syscalls *open*, *rename*, *truncate*, *unlink, removexattr*
FMT_REV.1	All modifications to the values of TSF data.	system calls *chmod*, *chown*, *setxattr*, *unlink*, *truncate*, *msgctl*, *semctl*, *shmctl*
FMT_SMF.1	None (covered by other management functions)	

Component	Event	Details (Event Names)
FMT_SMR.1	Modifications to the group of users that are part of a role.	Event: ``gpasswd:'' audit text messages ``group member added'', ``group member removed'', ``group administrators set'', ``group members set'' (from trusted programs in shadow suite).
FMT_SMR.1	Every use of the rights of a role. (Additional / Detailed)	The user's actions result in audited syscalls and the use of trusted programs that are audited. Details include the login ID, the origin can be determined from the associated LOGIN record for this login ID and audit session ID.
FPT_AMT.1	Execution of the tests of the underlying machine and the results of the test.	Event: ADMIN_amtu (generated by AMTU testing tool)
FPT_RVM.1	None	
FPT_SEP.1	None	
FPT_STM.1	Changes to the time.	Event: syscalls (do_)*settimeofday*, *adjtimex*
FTP_ITC.1	Set-up of trusted channel	Event: syscall *exec* (of `stunnel` program)

Application Note: The table lists the names of the events associated with the SFR. Details of the event specific data recorded with each event are defined in the audit design documentation.

5.1.1.2 User Identity Association (FAU_GEN.2)

FAU_GEN.2.1 The TSF shall be able to associate each auditable event with the identity of the user that caused the event.

Application Note (CAPP): There are some auditable events which may not be associated with a user, such as failed login attempts. It is acceptable that such events do not include a user identity. In the case of failed login attempts it is also acceptable not to record the attempted identity in cases where that attempted identity could be misdirected authentication data; for example when the user may have been out of sync and typed a password in place of a user identifier.

Rationale (CAPP): O.AUDITING calls for individual accountability (i.e., “TOE users”) whenever security-relevant actions occur. This component requires every auditable event to be associated with an individual user.

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

5.1.1.3 Audit Review (FAU_SAR.1)

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

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

Application Note (CAPP): The minimum information which must be provided is the same that which is required to be recorded in 5.1.1.2.

The intent of this requirement is that there exists a tool for administrator be able to access the audit trail in order to assess it. Exactly what manner is provided is an implementation decision, but it needs to be done in a way which allows the administrator to make effective use of the information presented. This requirement is closely tied to 5.1.5 and 5.1.6. It is expected that a single tool will exist within the TSF which will satisfy all of these requirements.

Rationale (CAPP): This component supports the O.AUDITING and O.MANAGE objectives by providing the administrator with the ability to assess the accountability information accumulated by the TOE.

Application Note: The TOE provides a tool “aucat” that transforms the audit records to human readable format.

5.1.1.4 Restricted Audit Review (FAU_SAR.2)

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

Application Note (CAPP): By default, authorized administrators may be considered to have been granted read access to the audit records. The TSF may provide a mechanism which allows other users to also read audit records.

Rationale (CAPP): This component supports the O.AUDITING objective by protecting the audit trail from unauthorized access.

Application Note: DAC controls ensure that only administrative users have access to the audit records.

5.1.1.5 Selectable Audit Review (FAU_SAR.3)

FAU_SAR.3.1 The TSF shall provide the ability to perform ***searches*** of audit data based on **the following attributes**:

a) User identity;

b) group identifier (real and effective)

c) event type

d) outcome (success/failure)

e) login from specific remote hostname

f) audit-id

g) process id

Application Note (CAPP): The ST must state the additional attributes that audit selectivity may be based upon (e.g., object identity, type of event), if any.

Rationale (CAPP): This component supports both the O.AUDITING and O.MANAGE objectives, by providing a means for the administrator to assess the accountability information associated with an individual user.

Application Note: The TOE provides a tool “augrep” that allows to filter the audit records according the criteria listed above.

5.1.1.6 Selective Audit (FAU_SEL.1)

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

a) User identity;

b) system call number

c) directory or file name.

Application Note (CAPP): The ST must state the additional attributes that audit selectivity may be based upon (e.g., object identity, type of event), if any.

Rationale (CAPP): This component supports both the O.AUDITING and O.MANAGE objectives, by providing a means for the administrator to assess the accountability information associated with an individual user.

Application Note: The TOE provides the administrator the ability to select the events to audit. This can be done by the administrator editing the filter configuration file of the audit daemon and then using the *auditd -r* command or the */etc/init.d/audit* script with the 'reload' parameter to notify the audit daemon of the change in the configuration. The audit daemon in turn notifies the kernel of the new auditing policy.

5.1.1.7 Guarantees of Audit Data Availability (FAU_STG.1)

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

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

Application Note (CAPP): On many systems, in order to reduce the performance impact of audit generation, audit records will be temporarily buffered in memory before they are written to disk. In these cases, it is likely that some of these records will be lost if the operation of the TOE is interrupted by hardware or power failures. The developer needs to document what the likely loss will be and show that it has been minimized.

Rationale (CAPP): This component supports the O.AUDITING objective by protecting the audit trail from tampering, via deletion or modification of records in it. Further it ensures that it is as complete as possible.

Application Note: This is achieved using the DAC controls.

5.1.1.8 Action in Case of Possible Audit Data Loss (FAU_STG.3)

FAU_STG.3.1 The TSF shall **generate an alarm to the authorized administrator** if the audit trail exceeds ***a value defined in the file audit.conf for the minimum space required for the file system the audit log file resides in***.

Application Note (CAPP): For this component, an "alarm" is to be interpreted as any clear indication to the administrator that the pre-defined limit has been exceeded. The ST author must state the pre-defined limit that triggers generation of the alarm. The limit can be stated as an absolute value, or as a value that represents a percentage of audit trail capacity (e.g., audit trail 75% full). If the limit is adjustable by the authorized administrator, the ST should also incorporate an FMT requirement to manage this function.

Rationale (CAPP): This component supports the O.AUDITING and O.MANAGE objectives by providing the administrator with a warning that a pending failure due to the exhaustion of space available for audit information.

Application Note: The alarm generated by the TOE is a syslog message. This message is generated when the audit trail capacity exceeds the limit defined in the audit.conf file. This limit can be defined by the system administrator by editing the audit.conf file and then re-boot the system.

The evaluated configuration uses bin-mode auditing, where a file roll-over strategy is used with several files. If one file gets full, the audit system switches to the next file and starts the "post-processing" program defined in the audit configuration. The post-processing program is responsible to save the data from the bin file so that it can be reused. This avoids the situation that the audit trail can ever reach a percentage of maximum space available and therefore the situation where this message is generated will not occur. When audit is used with file-mode auditing, the alarm as described in this requirement is generated.

5.1.1.9 Prevention of Audit Data Loss (FAU_STG.4)

FAU_STG.4.1 The TSF shall **be able to prevent auditable events, except those taken by the authorized administrator**, and ***stop all processes that attempt to generate an audit record*** if the audit trail is full.

Application Note (CAPP): The selection of "preventing" auditable actions if audit storage is exhausted is minimal functionality; providing a range of configurable choices (e.g., ignoring auditable actions and/or changing to a degraded mode) is allowable, as long as "preventing" is one of the choices. If configurable, then FMT_MOF.1 should be incorporated into the ST.

Rationale (CAPP): This component supports the O.AUDITING and O.MANAGE objectives by providing the audit trail is complete with respect to non-administrative users while providing administrators with the ability to recover from the situation.

Application Note: The TOE stops processes that want to generate an audit entry when the queue used for audit entries in the kernel is full. This queue will be continuously emptied by the audit daemon and the stopped processes will be resumed when there are empty entries in the queue. If the audit trail itself gets full, the audit daemon will not be able to empty the queue and therefore all processes that want to generate an audit record will be stopped. In the extreme case this would require the system administrator to re-boot the TOE in single user mode, back-up the audit trail and make space available for the audit trail and then restart the TOE.

5.1.2 Cryptographic Support (FCS)

5.1.2.1 Cryptographic key generation (SSL: Symmetric algorithms) (FCS_CKM.1(1))

FCS_CKM.1.1 The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm ***as defined in the SSL v3 standard*** and specified cryptographic key sizes ***128 bit (RC4), 168 bit (TDES), 128 bit (AES), 256 bit (AES)*** that meet the following: ***generation and exchange of session keys as defined in the SSL v3 and standard with the cipher suites defined in FCS_COP.1(2).***

Application Note: Generation of symmetric keys is defined in section 6.2 in the SSL v3 standard. The OpenSSL library used by the TOE also supports SSL v2, but this is seen as being not part of the evaluated configuration. The evaluation will assess that the keys are generated in accordance with the requirements defined in the SSL v3 standard. With respect to the strength of function, no assessment of the strength of the cryptographic algorithm itself and no analysis for potential weaknesses of keys with repect to the algorithm is performed. The key generation process will only be analysed and rated with respect to the entropy of the input to the key generation process and with respect to the fact that any postprocessing of this input will maintain the entropy.

5.1.2.2 Cryptographic key generation (SSH: Symmetric algorithms) (FCS_CKM.1(2))

FCS_CKM.1.1 The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm ***as defined in the SSH v2 standard SSH Transport Layer Protocol (draft-ietf-secsh-transport-15.txt)*** and specified cryptographic key sizes ***168 bit (TDES)*** that meet the following: ***generation and exchange of session keys as defined in the SSH v2 standard using the Diffie-Hellman key negotiation protocol.***

Application Note: For details of the key generation / key negotiation process see section 4.5, chapter 5 and chapter 6 of the SSH Transport Layer Protocol specification (draft-ietf-secsh-transport-15.txt) as published by the Secure Shell Charter of the Internet Engineering Task Force (IETF). The evaluation will assess that the keys are generated in accordance with the requirements defined in the SSH v2 standard. The key generation process will only be analysed and rated with respect to the entropy of the input to the key generation process and with respect to the fact that any postprocessing of this input will maintain the entropy.

5.1.2.3 Cryptographic key generation (SSL: RSA) (FCS_CKM.1(3))

FCS_CKM.1.1 The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm ***product specific*** and specified cryptographic key sizes ***1024 bit*** that meet the following: ***not specified***

Application Note: The SSL v3 specification does not define how the RSA key pair is generated. This is up to the implementation. Almost all implementations of the SSL v3 standard have their own algorithm for RSA key pair generation (if they support cipher suites that use RSA). Therefore the key generation and algorithm and the standard to follow are not defined here. Only the required key size is specified. The evaluation will assess that the keys generated form a correct RSA key pair. The key generation process will only be analysed and rated with respect to the entropy of the input to the key generation process and with respect to the primality tests and the probability of the numbers chosen to be prime.

5.1.2.4 Cryptographic key distribution (SSL: RSA public keys) (FCS_CKM.2(1))

FCS_CKM.2.1 The TSF shall distribute cryptographic keys in accordance with a specified cryptographic key distribution method ***digital certificates for public RSA keys*** that meets the following: ***certificate format as defined in the standard X.509 Version 3***.

Application Note: This requirement addresses the exchange of public RSA keys as part of the SSL client and server authentication.

5.1.2.5 Cryptographic key distribution (SSH: Diffie-Hellman key negotiation) (FCS_CKM.2(2))

FCS_CKM.2.1 The TSF shall distribute cryptographic keys in accordance with a specified cryptographic key distribution method ***diffie-hellman-group1-sha1*** that meets the following: ***Specification in Internet Draft: SSH Transport Layer Protocol (draft-ietf-secsh-transport-15.txt)***.

Application Note: The Diffie-Hellman protocol can be seen as a combined way to generate and distribute a shared session key between two communicating parties. So the Diffie-Hellman algorithm used by SSH is mentioned both in the key generation as well as in the key distribution security functional requirement.

5.1.2.6 Cryptographic key distribution (SSH: DSS public keys) (FCS_CKM.2(3))

FCS_CKM.2.1 The TSF shall distribute cryptographic keys in accordance with a specified cryptographic key distribution method ***digital certificates for public DSS keys*** that meets the following: ***ssh-dss key format as defined in: Internet Draft: SSH Transport Layer Protocol (draft-ietf-secsh-transport-15.txt)***.

5.1.2.7 Cryptographic key distribution (SSL: Symmetric keys) (FCS_CKM.2(4))

FCS_CKM.2.1 The TSF shall distribute cryptographic keys in accordance with a specified cryptographic key distribution method ***Secure Socket Layer handshake using RSA encrypted exchange of session keys*** that meets the following: ***SSL Version 3 (Internet Draft dated November 1996, Netscape Communication)***.

Application Note: This requirement addresses the exchange of SSL session keys as part of the SSL handshake protocol.

5.1.2.8 Cryptographic operation (RSA) (SSL: FCS_COP.1(1))

FCS_COP.1.1 The TSF shall perform ***digital signature generation and digital signature verification*** in accordance with a specified cryptographic algorithm ***RSA*** and cryptographic key sizes ***1024 bit*** that meet the following: ***SSL Version 3 (Internet Draft dated November 1996, Netscape Communication).***

Application Note: This requirement addresses the RSA digital signature generation and verification operations using the RSA algorithm as required by the SSL session establishment protocol (provided a cipher suite including RSA is used). Note that the details of the signature format like the use of the PKCS#1 block type 1 and block type 2 are defined in the SSL Version 3 standard.

5.1.2.9 Cryptographic operation (SSL: Symmetric operations) (FCS_COP.1(2))

FCS_COP.1.1 The TSF shall perform encryption and decryption in accordance with a specified cryptographic algorithm ***RC4, TDES and AES*** and cryptographic key sizes ***128 bit (RC4), 168 bit (TDES), 128 bit (AES) and 256 bit (AES)*** that meet the following: ***SSL Version 3 (Internet Draft dated November 1996, Netscape Communication) and the following cipher suites: SSL_RSA_WITH_RC4_128_SHA, SSL_RSA_WITH_3DES_EDE_CBC_SHA, as defined in the SSL v3 standard and TLS_RSA_WITH_AES_128_CBC_SHA, TLS_RSA_WITH_AES_256_CBC_SHA as defined in the IETF RFC 3268.***

5.1.2.10 Cryptographic operation (SSH: Symmetric operations) (FCS_COP.1(3))

FCS_COP.1.1 The TSF shall perform encryption and decryption in accordance with a specified cryptographic algorithm ***TDES*** and cryptographic key sizes ***168 bit (TDES)*** that meet the following: ***SSH Version 2 (Internet Draft: SSH Transport Layer Protocol (draft-ietf-secsh-transport-15.txt)) and the following cipher suite: 3des-cbc as defined in the SSH v2 internet draft mentioned above.***

5.1.3 User Data Protection (FDP)

5.1.3.1 Discretionary Access Control Policy (FDP_ACC.1)

FDP_ACC.1.1 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.**

Application Note CAPP): For most systems there is only one type of subject, usually called a process or task, which needs to be specified in the ST. Named objects are those objects which are used to share information among subjects acting on the behalf of different users, and for which access to the object can be specified by a name or other identity. Any object that meets this criterion but is not controlled by the DAC policy must be justified.

The list of operations covers all operations between the above two lists. It may consist of a sublist for each subject-named object pair. Each operation needs to specify which type of access right is needed to perform the operation; for example read access or write access.

Rationale (CAPP): This component supports the O.DISCRETIONARY_ACCESS objective by specifying the scope of control for the DAC policy.

5.1.3.2 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 ***filesystem*** **user identity and group membership(s) associated with a subject; and**

b) 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 filesystem 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 filesystem userid of the owner of the object but the filesystem 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 filesystem userid of the owner of the object nor is the filesystem 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 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 other than device special files on a file system mounted as read-only is always denied.
Write access to a file marked as immutable is always denied.

Application Note (CAPP): A CAPP conformant TOE is required to implement a DAC policy, but the rules which govern the policy may vary between TOEs; those rules need to be specified in the ST. In completing the rule assignment above, the resulting mechanism must be able to specify access rules which apply to at least any single user. This single user may have a special status such as the owner of the object. The mechanism must also support specifying access to the membership of at least any single group. Conformant implementations include self/group/public controls and access control lists.

A DAC policy may cover rules on accessing public objects; i.e., objects which are readable to all authorized users, but which can only be altered by the TSF or authorized administrators. Specification of these rules should be covered under FDP_ACF.1.3 and FDP_ACF.1.4.

A DAC policy may include exceptions to the basic policy for access by authorized administrators or other forms of special authorization. These rules should be covered under FDP_ACF.1.3.

The ST must list the attributes which are used by the DAC policy for access decisions. These attributes may include permission bits, access control lists, and object ownership.

A single set of access control attributes may be associated with multiple objects, such as all objects stored on a single floppy disk. The association may also be indirectly bound to the object, such as access control attributes being associated with the name of the object rather than directly to the object itself.

Rationale (CAPP): This component supports the O.DISCRETIONARY_ACCESS objective by defining the rules which will be enforced by the TSF.

5.1.3.3 Object Residual Information Protection (FDP_RIP.2)

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

Application Note (CAPP): This requirement applies to all resources governed by or used by the TSF; it includes resources used to store data and attributes. It also includes the encrypted representation of information.

Clearing the information content of resources on deallocation from objects is sufficient to satisfy this requirement, if unallocated resources will not accumulate new information until they are allocated again.

Rationale (CAPP): This component supports the O.RESIDUAL_INFORMATION objective.

5.1.3.4 Subject Residual Information Protection (Note 1)

NOTE 1 The TSF shall ensure that any previous information content of a resource is made unavailable upon the allocation of the resource to all subjects.

Application Note (CAPP): This requirement applies to all resources governed by or used by the TSF; it includes resources used to store data and attributes. It also includes the encrypted representation of information.

Clearing the information content of resources on deallocation from subjects is sufficient to satisfy this requirement, if unallocated resources will not accumulate new information until they are allocated again.

Rationale (CAPP): This component supports the O.RESIDUAL_INFORMATION objective.

5.1.3.5 Basic data exchange confidentiality (FDP_UCT.1)

FDP_UCT.1.1 The TSF shall enforce the ***Discretionary Access Control Policy*** to be able to ***transmit and receive*** objects in a manner protected from unauthorised disclosure.

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

5.1.3.6 Data exchange integrity (FDP_UIT.1)

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

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

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

5.1.4 Identification and Authentication (FIA)

5.1.4.1 User Attribute Definition (FIA_ATD.1)

FIA_ATD.1.1 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

Application Note (CAPP): The specified attributes are those that are required by the TSF to enforce the DAC policy, the generation of audit records, and proper identification and authentication of users. The user identity must be uniquely associated with a single individual user.

Group membership may be expressed in a number of ways: a list per user specifying to which groups the user belongs, a list per group which includes which users are members, or implicit association between certain user identities and certain groups.

A TOE may have two forms of user and group identities, a text form and a numeric form. In these cases there must be unique mapping between the representations.

Rationale (CAPP): This component supports the O.AUTHORIZATION and O.DISCRETIONARY_ACCESS objectives by providing the TSF with the information about users needed to enforce the TSP.

5.1.4.2 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.**

Application Note (CAPP): The method of authentication is unspecified by the CAPP, but must be specified in a ST. The method which is used must be shown to have low probability that authentication data can be forgotten or guessed. For example, if a password mechanism is used a set of metrics needs to be specified and may include such things as minimum length of the password, maximum lifetime of a password, and the subjecting of possible passwords to dictionary attacks. The strength of whatever mechanism implemented must be subjected to strength of function analysis. (See AVA_SOF.1)

Rationale (CAPP): This component supports the O.AUTHORIZATION objective by providing an authentication mechanism with a reasonable degree of certainty that only authorized users may access the TOE.

5.1.4.3 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 (CAPP): The ST must specify the actions which are allowed by an unauthenticated user. The allowed actions should be limited to those things which aid an authorized user in gaining access to the TOE. This could include help facilities or the ability to send a message to authorized administrators.

Rationale (CAPP): This component supports the O.AUTHORIZATION objective by specifying what actions unauthenticated users may perform.

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.

5.1.4.4 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.

Application Note (CAPP): Obscured feedback implies the TSF does not produce a visible display of any authentication data entered by a user, such as through a keyboard (e.g., echo the password on the terminal). It is acceptable that some indication of progress be returned instead, such as a period returned for each character sent.

Some forms of input, such as card input based batch jobs, may contain human readable user passwords. The Administrator and User Guidance documentation for the product must explain the risks in placing passwords on such input and must suggest procedures to mitigate that risk.

Rationale (CAPP): This component supports the O.AUTHORIZATION objective. Individual accountability cannot be maintained if the individual's authentication data, in any form, is compromised.

5.1.4.5 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.

Application Note (CAPP): The ST must specify the actions which are allowed to an unidentified user. The allowed actions should be limited to those things which aid an authorized user in gaining access to the TOE. This could include help facilities or the ability to send messages to authorized administrators.

The method of identification is unspecified by this PP, but should be specified in a ST and it should specify how this relates to user identifiers maintained by the TSF.

Rationale (CAPP): This component supports the O.AUTHORIZATION objective by specifying what actions unidentified users may perform.

5.1.4.6 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:

a) The user identity which is associated with auditable events;

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

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

d) ***no other security attributes*****.**

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 login user ID, the real user ID, the filesystem user ID and the effective user ID shall be those specified in the user entry for the user that has authenticated successfully.

b) Upon successful identification and authentication, the real group ID, filesystem group ID and the effective group ID 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 and filesystem 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 and filesystem user ID of the program owner. Access rights are then

evaluated using the filesystem user ID of the program owner. The real and login user ID remain unchanged.

b) ***The effective, filesystem and real user ID of a user can be changed by the su command. In this case the real, filesystem and effective user ID of the user is changed to the user specified in the su command (provided authentication is successful). The login user ID remains unchanged.***

c) ***The filesystem and 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 filesystem and effective group ID of the program owner. Access rights are then evaluated using the filesystem group ID of the program owner.***

Application Note (CAPP): The DAC policy and audit generation require that each subject acting on the behalf of users have a user identity associated with the subject. This identity is normally the one used at the time of identification to the system.

The DAC policy enforced by the TSF may include provisions for making access decisions based on a user identity which differs from the one used during identification.

The ST must state, in FIA_USB.1.1, how this alternate identity is associated with a subject and justify why the individual user associated with this alternate identity is not compromised by the mechanism used to implement it.

Depending on the TSF's implementation of group membership, the associations between a subject and groups may be explicit at the time of identification or implicit in a relationship between user and group identifiers. The ST must specify this association.

Like user identification, an alternate group mechanism may exist, and parallel requirements apply.

Rationale (CAPP): This component supports the O.DISCRETIONARY_ACCESS and O.AUDITING objectives by binding user identities to subjects acting on their behalf.

5.1.5 Security Management (FMT)

5.1.5.1 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 named 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.***

Application Note (CAPP): The ST must state the components of the access rights that may be modified, and must state any restrictions that may exist for a type of authorized user and the components of the access rights that the user is allowed to modify.

The ability to modify access rights must be restricted in that a user having access rights to a named object does not have the ability to modify those access rights unless granted the right to do so. This restriction may be explicit, based on the object ownership, or based on a set of object hierarchy rules.

Rationale (CAPP): This component supports the O.DISCRETIONARY_ACCESS objective by providing the means by which the security attributes of objects are managed by a site.

5.1.5.2 Secure security attributes (FMT_MSA.2)

FMT_MSA.2.1 The TSF shall ensure that only secure values are accepted for security attributes.

Application Note: This requirement is included as a dependency from the security functional requirements FCS_CKM.1, FCS_CKM.2 and FCS_COP.1. The assessment with respect to this requirement in the evaluatiuon of this TOE does not include any assessment of the cryptographic strength of the keys generated or used. Instead the assessment with respect to this requirement just includes an assessment that the TOE protects those keys from unauthorized access, disclosure or tampering.

5.1.5.3 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 (CAPP): A CAPP-conformant TOE must provide protection by default for all objects at creation time. This may be done through the enforcing of a restrictive default access control on newly created objects or by requiring the user to explicitly specify the desired access controls on the object at its creation. In either case, there shall be no window of vulnerability through which unauthorized access may be gained to newly created objects.

Rationale (CAPP): This component supports the O.DISCRETIONARY_ACCESS objective by requiring that objects are properly protected starting from the instant that they are 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.

5.1.5.4 Management of the Audit Trail (FMT_MTD.1)

FMT_MTD.1.1 The TSF shall restrict the ability to **create, delete, and clear** the **audit trail** to **authorized administrators**.

Application Note (CAPP): The selection of "create, delete, and clear" functions for audit trail management reflect common management functions. These functions should be considered generic; any other audit administration functions that are critical to the management of a particular audit mechanism implementation should be specified in the ST.

Rationale (CAPP): The component supports the O.AUDITING and O.MANAGE objectives by ensuring that the accountability information is not compromised by destruction of the audit trail.

Application Note: This requirement is implemented using the discretionary access control features of the TOE to protect the files holding the audit trail.

5.1.5.5 Management of Audited Events (FMT_MTD.1)

FMT_MTD.1.1 The TSF shall restrict the ability to **modify or observe** the **set of audited events** to **authorized administrators**.

Application Note (CAPP): The set of audited events are the subset of auditable events which will be audited by the TSF. The term set is used loosely here and refers to the total collection of possible ways to control which audit records get generated; this could be by type of record, identity of user, identity of object, etc.

It is an important aspect of audit that users not be able to effect which of their actions are audited, and therefore must not have control over or knowledge of the selection of an event for auditing.

Rationale (CAPP): This component supports the O.AUDITING and O.MANAGE objectives by providing the administrator with the ability to control the degree to which accountability is generated.

Application Note: This requirement is implemented using the discretionary access control features of the TOE to protect the audit configuration files.

5.1.5.6 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 **authorized administrators**.

Application Note (CAPP): This component only applies to security attributes which are used to maintain the TSP. Other user attributes may be specified in the ST, but control of those attributes is not within the scope of the CAPP.

Rationale (CAPP): This component supports the O.MANAGE objective by providing the administrator with the means to manage who are authorized users and what attributes are associated with each user.

5.1.5.7 Management of Authentication Data (FMT_MTD.1)

FMT_MTD.1.1 The TSF shall restrict the ability to **initialize** the **authentication data** to **authorized administrators**.

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

a) authorized administrators; and

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

Application Note (CAPP): User authentication data refers to information that users must provide to authenticate themselves to the TSF. Examples include passwords, personal identification numbers, and fingerprint profiles. User authentication data does not include the user’s identity. The ST must specify the authentication mechanism that makes use of the user authentication data to verify a user’s identity.

This component does not require that any user be authorized to modify their own authentication information; it only states that it is permissible. It is not necessary that requests to modify authentication data require reauthentication of the requester’s identity at the time of the request.

Rationale (CAPP): This component supports the O.AUTHORIZATION and O.MANAGE objectives by ensuring integrity and confidentiality of authentication data.

5.1.5.8 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 **authorized administrators**.

FMT_REV.1.2 The TSF shall enforce the rules:

a) The immediate revocation of security-relevant authorizations; and

b) Revocations/modifications made by an authorized administrator 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 (CAPP): 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. The immediate method must be specified in the ST and in administrator guidance. In a distributed environment the developer must provide a description of how the “immediate” aspect of this requirement is met.

Rationale (CAPP): This component supports the O.MANAGE objective by controlling access to data and functions which are not generally available to all users.

Application Note: Like other UNIX type operating systems also the TOE does not enforce “immediate revocation” for user security attributes. To achieve this the system administrator has to check, if the user whose security attributes have been changed is currently logged in. If this is the case, the system administrator has to “force” the user to log off as indicated in the CAPP Application Note.

5.1.5.9 Revocation of Object Attributes (FMT_REV.1)

FMT_REV.1.1 The TSF shall restrict the ability to revoke security attributes associated with **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) **The access rights associated with an object shall be enforced when an access check is made; and**

b) ***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.***

Application Note (CAPP): The DAC policy may include immediate revocation (e.g., Multics immediately revokes access to segments) or delayed revocation (e.g., most UNIX systems do not revoke access to already opened files). The DAC access rights are considered to have been revoked when all subsequent access control decisions by the TSF use the new access control information. It is not required that every operation on an object make an explicit access control decision as long as a previous access control decision was made to permit that operation. It is sufficient that the developer clearly documents in guidance documentation how revocation is enforced.

Rationale (CAPP): This component supports the O.DISCRETIONARY_ACCESS objective by providing that specified access control attributes are enforced at some fixed point in time.

Application Note: Like most other UNIX type operating systems the TOE implements delayed revocation as indicated in the CAPP Application Note.

5.1.5.10 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**
- **Audit event 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.

5.1.5.11 Security Management Roles (FMT_SMR.1)

FMT_SMR.1.1 The TSF shall maintain the roles:

a) **authorized administrator;**

b) **users authorized by the Discretionary Access Control Policy to modify object security attributes;**

c) **users authorized to modify their own authentication data; and**

d) ***no other roles***

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

Application Note (CAPP): A CAPP-conformant TOE only needs to support a single administrative role, referred to as the authorized administrator. If a TOE implements multiple independent roles, the ST should refine the use of the term authorized administrators to specify which roles fulfill which requirements.

The CAPP specifies a number of functions which are required of or restricted to an authorized administrator, but there may be additional functions which are specific to the TOE. This would include any additional function which would undermine the proper operation of the TSF. Examples of functions include: ability to access certain system resources like tape drives or vector processors, ability to manipulate the printer queues, and ability to run real-time programs.

Rationale (CAPP): This component supports the O.MANAGE objective.

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.

5.1.6 Protection of the TOE Security Functions (FPT)

5.1.6.1 Abstract Machine Testing (FPT_AMT.1)

FPT_AMT.1.1 The TSF shall run a suite of tests ***at the request of an authorized administrator*** to demonstrate the correct operation of the security assumptions provided by the abstract machine that underlies the TSF.

Application Note (CAPP): In general this component refers to the proper operation of the hardware platform on which a TOE is running. The test suite needs to cover only aspects of the hardware on which the TSF relies to implement required functions, including domain separation. If a failure of some aspect of the hardware would not result in the TSF compromising the functions it performs, then testing of that aspect is not required.

Rationale (CAPP): This component supports the O.ENFORCEMENT objective by demonstrating that the underlying mechanisms are working as expected.

Application Note: The abstract machine testing tool will be platform dependent. Chapter 6 describes the common feature of all those tools. The reader should be aware that in the case of xSeries and pSeries the abstract machine is the real hardware, while in the case of zSeries, iSeries and eServer the abstract machine is a virtualization of the real hardware by a logical partitioning layer.

5.1.6.2 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.

Application Note (CAPP): This element does not imply that there must be a reference monitor. Rather this requires that the TSF validates all actions between subjects and objects that require policy enforcement.

Rationale (CAPP): This component supports O.ENFORCEMENT objective by ensuring that the TSP is not being bypassed.

5.1.6.3 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.

Application Note (CAPP): This component does not imply a particular implementation of a TOE. The implementation needs to exhibit properties that the code and the data upon which TSF relies are not alterable in ways that would compromise the TSF and that observation of TSF data would not result in failure of the TSF to perform its job. This could be done either by hardware mechanisms or hardware architecture. Possible implementations include multi-state CPU's which support multiple task spaces and independent nodes within a distributed architecture.

The second element can also be met in a variety of ways also, including CPU support for separate address spaces, separate hardware components, or entirely in software. The latter is likely in layered application such as a graphic user interface system which maintains separate subjects.

Rationale (CAPP): This component supports O.ENFORCEMENT objectives by ensuring that a TSF exists within the TOE and that it can reliably carry out its functions.

Application Note: The TOE enforces this requirement by using the address separation features provided by the Memory Management Units and the protection offered by a multi-state CPU. Although the TOE operates on four different platforms, all those platforms have in common a Memory Management Unit allowing to define address space separation between trusted and untrusted subjects and all platforms support a multi-state CPU where modification to the address space definition and direct access to peripheral devices and the CPU configuration can be restricted to a state reserved for a defined part of the TSF (the kernel). The TOE ensures that those features are used correctly to prohibit any untrusted subject from unallowed interference and tampering with the TSF.

5.1.6.4 Reliable Time Stamps (FPT_STM.1)

FPT_STM.1.1 The TSF shall be able to provide reliable time stamps for its own use.

Application Note (CAPP): The generation of audit records depends on having a correct date and time. The ST needs to specify the degree of accuracy that must be maintained in order to maintain useful information for audit records.

Rationale (CAPP): This component supports the O.AUDITING objective by ensuring that accountability information is accurate.

Application Note: The TOE uses a hardware timer to maintain its own time stamp. This hardware timer is protected from tampering by untrusted subjects. The start value for this timer may be set by the system administrator, but the system administrator may also start a program that uses an external trusted time source to set this initial value.

5.1.6.5 Inter-TSF trusted channel (FTP_ITC.1)

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

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

FTP_ITC.1.3 The TSF shall initiate communication via the trusted channel for ***when the communication uses the SSH v2 or SSL v3 protocol offered as services by the TOE***.

5.1.7 Strength of Function

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

Note: The security functions within the TOE that uses a permutational or probabilistic mechanism are the authentication function that uses passwords and the random number generation process used as input for the process to generate cryptographic keys. For RSA key pairs also the probability of the numbers used to be prime is assessed. No strength of function analysis is performed for the cryptographic algorithms themselves which also excludes any analysis of the existence and characterization of cryptographically weak keys. This statement is made in compliance with part 1 of the CC and paragraph 424 of part 2 of the CEM.

5.2 TOE Security Assurance Requirements

The target evaluation assurance level for the product is EAL4 [CC] augmented by ALC_FLR.3.

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 defines conditions (when the processor encounters an exception, an interrupt or when a call gate for a higher ring of privilege is called. The security requirement FMT_MSA.1 was therefore not applicable because the security attribute 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 9 with the certification-sles-ibm-eal4 package that are subject to this evaluation. A large subset of the overall security related functions of SLES 9 has been included in this evaluation. Those functions provide the basic security for a server within a protected environment. They allow for identification and authentication of users, access control to files and IPC objects, auditing of security critical events and the secure communication with other trusted systems. The TOE protects the security functions from unauthorized tampering and bypassing and allows only administrative users to manage the security functions. Normal users are only allowed to manage access control rights of the file system and IPC objects they own and to modify their own password in accordance with the password rules enforced by the TOE. Those functions are required as a basis for application level security functions and mechanisms and can be used to build application specific security policies.

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 six different types of file systems included in the evaluated configuration: the journalled file system ext3, CDROM File System ISO-9660 (read-only), the proc file system, the sysfs file system, the tmpfs file system and the devpts file system. ext3 and ISO-9660 represent file systems on a physical medium (disk, CDROM). The proc and the sysfs file systems do not represent or provide a physical data storage file system but are 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. The sysfs file system is a new harmonized interface to non-process related kernel information (mainly for device drivers). Access to objects there can be restricted using the DAC mechanism (which are the permission bits only).

The tmpfs file system implements a fast, RAM based file system used for fast access to temporary files. This file system is not persistent across boots of the operating system.

The devpts file system provides pseudo terminal support.

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 TCP based application protocols supporting user authentication and running on privileged ports are:

- secure shell (SSH v2)
- file transfer services (FTP)

In addition the TOE supports secure socket layer (SSL v3) protocol, which can be used to securely tunnel higher layer protocols. This service is provided by a trusted process which can be used by applications to tunnel TCP based protocols using a single port. The tunnel actually provides the certificate based authentication of the server side of the tunnel and the confidentiality and integrity protection of the communication.

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 networked 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 perform security functions and therefore part of the TSF interface.
- Network Services, (ssh, ftp, ssl). 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 transferring 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, ssl and ftp are seen as TSF, because they use privileged ports. ssh and ftp require user identification and authentication and ssh and ssl provide confidentiality and integrity protection

  Note: Users may start programs using unprivileged ports, but those programs operate with the effective and filesystem 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 synchronization 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 synchronize 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
- Audit

- Discretionary Access Control
- Object Reuse
- Security Management
- Secure Communication
- 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.

The module pam_passwdqc.so provides additional checks for the strength of passwords, allowing for a more strict password policy.

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 synchronized 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 synchronizing 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* contains 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/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.

The time of the last successful logins is recorded in */var/log/lastlog* (IA1.6).

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/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: 128 (IA1.7)

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 real, filesystem and 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, filesystem and effective user ID and real, filesystem 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, filesystem and effective user ID to root even if they would know the authentication information for root. Note that when a user executes a program that has the setuid bit set only the effective user ID and the filesystem user ID are 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 login ID is neither changed by the su command nor by executing a program that has the setuid or setgid bit set (IA4.6).

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 login, real, filesystem and effective user ID are set to the ID of the user that has logged in (IA5.1). With the su command the real, filesysetm and the effective user ID and the real, filesystem and the effective group ID are changed but the login ID remains unchanged.

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

6.2.3 Audit (AU)

The Linux Audit Subsystem (LAuS) is designed to be a CAPP compliant audit system for Linux. LAuS is built on top of systrace which is a system call security policy enforcement engine first developed for BSD but ported to

Linux. The subsystem allows configuring the events to be actually audited from the set of all events that are possible to be audited. Those events are configured in a specific configuration file and then the kernel is notified to build its own internal structure for the events to be audited.

6.2.3.1 Audit Configuration (AU.1)

The system administrator can define the events to be audited from the overall events that the Linux Audit Subsystem is able to audit using rules defined in the *filter.conf* audit configuration file using predicates and logical operations (AU1.1). This allows for a very flexible definition of the events to be audited and the conditions under which events are audited. The system administrator is also able to define a set of user IDs for which auditing is active (AU1.2) or alternatively a set of user IDs that are not audited (AU1.3). Changes to the audit configuration take effect when the audit daemon is notified about a change in the audit configuration (AU1.4).

This notification can only be performed by an administrative user (using the *auditd -r* command or the */etc/init.d/audit script* with the 'reload' parameter) (AU1.5).

This function contributes to satisfy the security requirements FAU_SEL.1 and FMT_MTD.1 (Management of audited events)

6.2.3.2 Audit Processing (AU.2)

Auditing is performed on a per process basis. A process can enable or disabling auditing for itself by attaching itself or detaching itself to the audit subsystem provided it is running with root privileges (AU2.1). The attribute of being attached to the audit subsystem is inherited by all processes that are forked off from a process, which ensures that events generated by child processes are also audited (AU2.2).

The kernel audits system calls in accordance with the rules defined in the *filter.conf* audit configuration file. In addition trusted processes can generate audit records and send them to the kernel (AU2.3). The login ID is associated with audit events ensuring that events can be easily associated with the ID a user used to log into the TOE (AU2.4).

The events to be audited are forwarded by the kernel to an audit daemon, which writes the audit records to the audit trail. An internal queuing mechanism is used for this purpose. When the queue does not have sufficient space to hold an audit record a process wants to create, the process that wants to generate the audit record is halted until the queue has enough space again (AU2.5). This ensures that audit records do not get lost due to resource shortage.

The audit daemon has three ways in which it can write the audit records to disk. The choice of which method to use is configurable by the administrator. The three choices are bin mode, file mode and stream mode. In stream mode, an audit record stream is piped to a user defined program for post-processing. In file mode audit records are appended to a defined file. In bin mode, several fixed length files are maintained with a pointer to the current location. The audit records are written until the current file has reached it maximum capacity and then the next file is utilized until the last file reaches its maximum capacity at which point the first file is used again (AU2.6). Whenever such a switch between two files happens the audit subsystems starts a program defined in the audit configuration file to process the data collected in the audit bin file (in the evaluated configuration this program is the *aucat* program that converts the binary data into a human readable format and appends the result to an existing audit data file) (AU2.7). In the evaluated configuration bin mode auditing has to be used (since the post-processing program required for stream mode would be part of the TOE and required to be evaluated).

The audit configuration file contains a parameter indicating the "fill level" of the audit trail that causes a warning message to be written to syslog. This warning is only useful and therefore only generated when file-mode auditing is selected. This is used to inform the system administrator that he needs to back-up the current audit trail and make space available for additional audit records. In the case the system administrator does not perform this in time and the audit trail gets full, the audit daemon will not be able to collect audit records from the internal queue and therefore any process that is going to create a new audit record will be halted. In the extreme case the system administrator will need to shut down the TOE, restart it in single-user mode to back-up and clear the audit trail and then re-boot the TOE in secure mode. This situation will not occur when bin-mode auditing is selected (as is the mode for the evaluated configuration).

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

This function contributes to satisfy the security requirements FAU_SAR.2, FAU_STG.1, FAU_STG.3, FAU_STG.4 and FMT_MTD.1 (Management of the audit trail).

6.2.3.3 Audit Record Format (AU.3)

An audit record consists of a standard header common to all audit events followed by event specific data. The standard header contains the following information:

- Audit ID: unique 32 bit identifier
- Login ID: User ID of the user authenticated by the system (regardless if the user has changed his real and / or effective user ID afterwards)
- Effective user ID: the effective user ID of the process at the time the audit event was generated
- Timestamp: Date and time the audit record was generated
- The type of the event
(AU3.1)

Note: Although the TOE distinguishes between the effective and the filesystem user ID, those two are identical in all states of the TOE.

The event specific data will always contain data indicating if the request that caused the event has been successful or not (AU3.2).

The audit subsystem maintains a “Login ID” which is set when the user performs his initial login at a terminal or via a network connection (AU3.3). This Login ID is maintained for actions of this user until he terminates the session. This Login ID remains unchanged when the user performs a switch of the real and / or effective and filesystem user ID by the su command or by invoking a program that has the setuid bit set (AU3.4). This allows to trace all actions to the real user.

This function contributes to satisfy the security requirements FAU_GEN.1 and FAU_GEN.2

6.2.3.4 Audit Post-Processing (AU.4)

The TOE provides two tools for the post-processing of audit data:

aucat reads the raw binary audit data and transforms it into human readable format (AU4.1).

augrep allows to selectively extract records from the audit trail using defined selection criteria (AU4.2).

This function contributes to satisfy the security requirements FAU_SAR.1 and FAU_SAR.3.

6.2.4 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.

6.2.4.1 General DAC Policy (DA.1)

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 a filesystem user ID of 0 is exempt from all restrictions and can perform any action desired (DA1.1).

DAC provides the mechanism that allows users to specify and control access to objects that they own (DA1.2). DAC attributes are assigned to objects at creation time and remain in effect until the object is destroyed or the object attributes are changed (DA1.3). 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 filesystem 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) (DA1.4). Changes to the file group are restricted to the owner and root (DA1.5).

The new file group identifier must either be the current filesysteme group identifier or one of the group identifiers in the concurrent group set (DA1.6). In addition, a subject whose filesystem 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 DA1.1).

Permission bits are the standard UNIX DAC mechanism and are used on all SLES file system named objects (DA1.7). 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 (DA1.8).

Write access is in general not granted for files on a file system mounted as read-only (DA1.9). Write access is also denied for files that have the immutable attribute (DA1.10).

6.2.4.2 Permission Bits (DA.2)

SLES supports standard UNIX permission bits to provide one form of DAC for file system objects in the proc, sysfs, tmpfs, devpts 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. Note also that access to specific objects in the /proc file system may be restricted to root regardless of the setting of the permission bits.

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
  (DA2.1)

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

- Users with a filesystem user ID of 0 are able to read and write all files, ignoring the permission bits. Users with a filesystem user ID of zero are also able to execute any file if it is executable for someone.
- If the filesystem 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 filesystem 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.
  (DA2.2)

This function contributes to satisfy the security requirements FAU_SAR.2, FDP_ACC.1 and FDP_ACF.1.

6.2.4.3 Access Control Lists supported by SLES (DA.3)

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 the 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
(DA3.1)

6.2.4.3.1 ACL Tag Types

The following tag types exist:

1. ACL_GROUP
an ACL entry of this type defines access rights for processes whose filesystem 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 filesystem 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 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 filesystem user ID matches the ACL entry qualifier
6. ACL_USER_OBJ
an ACL entry of this type defines access rights for processes whose filesystem user ID matches the user ID of the owner of the file
(DA3.2)

6.2.4.3.2 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 (DA3.3).

6.2.4.3.3 ACL Permissions

The permission that can be defined in an ACL entry are: read, write and execute/search (DA3.4).

6.2.4.3.4 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") (DA3.5). An ACL may have between zero and a defined maximum number of entries of the type ACL_GROUP and ACL_USER (DA3.6).

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 a filesystem user ID and filesystem group ID or supplementary group ID different from those defined in ACL_USER_OBJ and ACL_GROUP_OBJ entries (DA3.7).

6.2.4.3.5 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 (DA3.8).

6.2.4.3.6 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 (DA3.9).

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.

6.2.4.3.7 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 for file system objects other than device special files.

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

3. **If** the filesystem 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 filesystem 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 filesystem 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.

(DA3.10)

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

6.2.4.3.8 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 (DA3.11).

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.

6.2.4.3.9 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 (DA3.12).

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 (DA3.13).

6.2.4.3.10 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 (DA3.14).

UNIX domain sockets exist in the file system name space, the socket files can have both base mode bits and extended ACL entries (DA3.15).

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 (DA3.16).

6.2.4.3.11 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 (DA3.17). 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).

6.2.4.3.12 DAC: Device Special File

The access control scheme described for file system objects is used for protection of character and block device special files (DA3.18). 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 (DA3.19). 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 (DA3.20).

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

6.2.4.3.13 DAC: SYSFS File System

The 2.6 version of the Linux kernel has introduced a new file system to assist in the handling of device driver charcteristics. The sysfs file system uses the permission bits for access control and protection and therefore access control follows the semantics of the permission bits (DA3.21). A specific type of access requires that a method to handle this access has been registered with the object when it was created in the sysfs file system (DA3.22).

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

6.2.4.4 Discretionary Access Control: IPC Objects (DA.4)

6.2.4.4.1 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 (DA4.1).

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) (DA4.2).

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 (DA4.3).

- The owning user and creating user of a newly created SMS will be the effective user ID of the creating process (DA4.4).
- The owning group and creating group of a newly created SMS will be the effective group ID of the creating process (DA4.5).

- 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 (DA4.6).
- SMSs do not have ACLs as described above, they only have permission bits (DA4.7).

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 (DA4.8). Access permissions can also be changed by any process with an effective user ID of 0, also known as running with the root identity (DA4.9).

6.2.4.4.2 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) (DA4.10). Changes to access controls (i.e., revocation) are effective upon the next request for access (DA4.11). 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 (DA4.12).

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) (DA4.13). However, once a message queue has been marked as deleted, additional processes cannot perform messaging operations and it cannot be undeleted (DA4.14).

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.
  (DA4.15)

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 (DA4.16).

6.2.4.4.3 DAC: Semaphores

For semaphores, access checks are performed for each access request (e.g., to lock or unlock the semaphore) (DA4.17). Changes to access controls (i.e., revocation) are effective upon the next request for access (DA4.18). 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 (DA4.19).

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 Evaluated Configuration Guide [SECGUIDE]. 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) (DA4.20). However, once a semaphore has been marked as deleted, additional processes cannot perform semaphore operations and it cannot be undeleted (DA4.21).

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 (DA4.22).

- 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 (DA4.23).

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 (DA4.24). Access permissions can also be changed by any process with an effective user ID of 0 (DA4.25).

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

6.2.5 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.

6.2.5.1 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 Evaluated Configuration Guide not to mount file systems on these devices.

Object reuse in the tmpfs file system is handled by the memory management object reuse functions. When allocating new space for a file, the TOE uses the functions of the memory management which clear the memory before it is allocated.

Object reuse for objects in the devpts file system are handled by the VFS layer. Note that devpts is not a disk based file system and therefore object reuse of disk space is not an issue for this file system.

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.

6.2.5.2 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.

6.2.5.3 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 and Note 1.

6.2.6 Security Management (SM)

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

6.2.6.1 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 configuration 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.

6.2.6.1.1 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).

6.2.6.1.2 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.

6.2.6.2 Access Control Configuration and Management (SM.2)

Access control to objects is defined by the permission bits or by the Access Control Lists (for those objects that have access control lists associated 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".

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

6.2.6.3.1 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 files */etc/passwd* and */etc/groups*):

- 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* and */etc/groups* (for the list of all groups the user belongs to). (SM3.2)

6.2.6.3.2 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/passwd*) (SM3.3).

6.2.6.3.3 Management of Authentication Data

An administrative user has the capability to define rules and restrictions for passwords used to authenticate users. The parameters 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".

6.2.6.4 Management of Audit Configuration (SM.4)

The TOE allows configuring the events to be audited. Those events are defined in a specific configuration file and then the *auditd -r* command (or the */etc/init.d/audit* script with the 'reload' parameter) is used to notify the audit subsystem about modifications in the rules defining the events to be audited. The use of the *auditd* command and the */etc/init.d/audit* script is restricted to administrative users. In addition the TOE allows an administrative user to start or stop the audit subsystem (also using the */etc/init.d/audit* script to start the audit subsystem (using the 'start" parameter) or stop the audit subsystem (using the 'stop' parameter) (SM4.1).

The administrative user can define the events to be audited in form of a set of rules using predicates and logical operations (SM4.2).

This function contributes to satisfy the security requirements FAU_GEN.1 and FAU_SEL.1 as well as FMT_MTD.1 (Management of the audit trail) and FMT_MTD.1 (Management of audited events)

6.2.6.5 Reliable Time Stamps (SM.5)

The TOE maintains a reliable clock used to generate time stamps as required for the TOE itself and applications. The audit subsystem requires such a reliable time source for the date and time field in the header of each audit record. The clock uses timers provided by the hardware and interrupt routines that update the value of the clock maintained by the TOE.

The initial value for this clock may be provided by a hardware clock that is part of the TOE hardware, by a trusted external time source (e. g. via the ntp protocol) or by the system administrator setting the initial value. Only the system administrator is allowed to overwrite the value of the clock maintained by the TOE (e. g. to correct the value in case it has drifted over time due to some inaccuracy of the hardware timer used by the TOE) (SM5.1).

This function contributes to satisfy the security requirement FPT_STM.1

6.2.7 Secure Communication (SC)

The TOE provides the ability to protect communication by cryptographic mechanism against disclosure and undetected unauthorized modification. The TOE supports two protocols (SSH v2 and SSL v3) that provide protection of communication against the above mentioned threats. **Note that communication using other protocols is not protected against those threats.**

The protocols SSH v2 and SSL v3 allow a secure communication between the TOE and a remote trusted IT product (which may be another instantiation of the TOE itself) over an insecure network. Within the TOE the protocols are configured to allow the secure tunneling of TCP based protocols. The difference between the two possibilities for tunneling consists in the authentication involved.

In the case of the SSH protocol the TOE supports establishing a secure connection allowing an application on a client system to set up the communication to the server side system after successful user authentication. This allows to get access to a shell from a remote system but also to perform actions like secure file transfer where access to the files on the remote system is protected by the discretionary access control mechanism.

In the case of the SSL protocol, the TOE would allow to set up a secure communication channel between a client and an untrusted application (e. g. a web server) on the server side. This would allow a client to access the web server without user authentication but (depending on the configuration of the SSL server) with the certificate based authentication of the client system.

6.2.7.1 Secure Protocols (SC.1)

The TOE offers two protocols that applications can use to securely communicate with another trusted IT product (provided this supports those protocols in the same way as the TOE does). Those protocols are the Secure Shell Protocol Version 2 (SSH v2) and the Secure Socket Layer Protocol Version 3 (SSL v3) (SC1.1). Both protocols are able to establish a secure channel between a client and a server process. The TOE supports both the client as well as the server processes for both of those protocols and therefore is able to initiate a connection as well as act as the receiver part. Both protocols provide the ability to “tunnel” a otherwise unprotected single port TCP based protocol.

6.2.7.1.1 The Secure Shell Protocol

The TOE provides the Secure Shell Protocol Version 2 (SSH v2) to allow users from a remote host to establish a secure connection and perform a logon to the TOE. The TOE supports the following security functions of the SSH v2 protocol:

1. Establishing a secure communication channel using the following cryptographic functions provided by the SSH v2 protocol:
   - o Encryption using three key Triple DES in CBC mode (3des-cbc as defined in section 4.3 of [SSH-TRANS]) (SC1.2)
   - o Diffie-Hellman key exchange (diffie-hellman-group1-sha1 as defined in section 6.1 of [SSH-TRANS]) (SC1.3)
   - o The keyed hash function hmac-sha1 for integrity protection as defined in section 4.4 of [SSH-TRANS] (which refers to [RFC2104] for the exact definition of the algorithm) (SC1.4).

   Note: The protocol supports more cryptographic algorithms than the ones listed above. Those other algorithms are not covered by this evaluation and shall be disabled when running the evaluated configuration.

2. Performing user authentication using the standard password based authentication method the TOE provides for users (Password Authentication Method as defined in chapter 5 of [SSH-AUTH]) (SC1.5).

   Note: The protocol also supports other authentication methods (e. g. certificate based authentication) but those are not within the scope of this Security Target. This Security Target requires password based authentication and therefore the SSH v2 server should be configured to accept this authentication method only.

3. Checking the integrity of the messages exchanged and close down the connection in case an integrity error is detected (SC1.6).

6.2.7.1.2 The Secure Socket Layer Protocol

The TOE provides the Secure Socket Layer Protocol Version 3 (SSL v3) to allow users from a remote host to establish a secure channel to the TOE. In contrast to the Secure Shell protocol described above, the SSL protocol does not support user authentication as part of the protocol. The SSL protocol within the TOE also allows to tunnel other TCP based protocols (that satisfy the restrictions defined in the Evaluated Configuration Guide) securely between a client and a server system.

On the client as well as on the server side the Stunnel program can be used to tunnel non-SSL aware daemons and protocols (like POP, IMAP, LDAP, etc) by having Stunnel provide the encryption, requiring no changes to the daemon's code. Stunnel acts as a trusted wrapper that can be used by applications implementing otherwise non-secure protocols. Stunnel as part of the TSF will ensure that the user data transmitted by those applications over the network will be confidentiality and integrity protected by the SSL v3 protocol. For guidance on how to set up such trusted channel and how to use it by applications please see the Evaluated Configuration Guide [SECGUIDE].

The Stunnel daemon will be configured to support the following cypher suites defined in the SSL v3 protocol or RFC3268:

SSL_RSA_WITH_RC4_128_SHA

SSL_RSA_WITH_3DES_EDE_CBC_SHA

TLS_RSA_WITH_AES_128_CBC_SHA

TLS_RSA_WITH_AES_256_CBC_SHA (SC1.7)

Other cypher suites as defined in the SSL v3 specification or RFC3268 are not supported in this Security Target and the TOE should be configured to not support other cipher suites.

This implies that the following cryptographic algorithms from the OpenSSL library are used:

1. The RSA algorithm with 1024 bit modulus length. RSA is used for the exchange of the session key and for server authentication.
2. RC4 with a key size of 128 bit (as one alternative for the symmetric encryption algorithm)
3. Triple DES with a key size of 168 bit
4. AES with a key size of 128 or 256 bit
5. SHA-1 (as the cryptographic hash function)

An implication of the use of this cipher suite and its algorithms is the authentication of the SSL server site using digital certificates.

Note: The function to generate the RSA key pair used by the server is part of the TSF, but the generation of the certificate of the public key is regarded as an aspect of the IT environment. A widely accepted Certification Authority might be used to generate this certificate (allowing a wide community trusting this CA to validate the certificate). In a closed community it might also be sufficient to have one server within the community to act as a CA. The OpenSSL library provides the functions to set up such a CA, but those functions are not subject of this Security Target.

This function contributes to satisfy the security requirements FCS_CKM.1 (1-3), FCS_CKM.2 (1-4), FCS_COP.1 (1-3), FDP_UCT.1, FDP_UIT.1, FMT_MSA.2 and FTP_ITC.1.

6.2.8 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. 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 and main storage defined by the kernel to be directly accessible by a user process.

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

6.2.8.1 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.

6.2.8.2 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 kernel privileges but only defined subsystems within the kernel are part of the TSF. The kernel is entered by some event that causes a context switch such as a system call, I/O interrupt, or a program exception condition.

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

6.2.8.3 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/modprobe.conf".

- modprobe is then asked to insert the module(s) that it has decided that the kernel needs. Every module will be configured according to the "options" lines in "/etc/modprobe.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 explicitly.

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

6.2.8.4 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 check for each access to an object protected by the any of DAC mechanism, 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.

6.2.8.5 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. Synchronizing those databases is not performed in the evaluated configuration. If such synchronization is required by an organization it is the responsibility of an administrative user of the TOE to achieve this either manually or with some automated assistance.

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/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/modprobe.conf	Configuration file for modprobe. modprobe automatically loads or unloads a module while taking into account its dependencies.
/etc/modprobe.conf.local	Local modprobe component configuration file.
/etc/modprobe.d	Contains 2.6 format component configurations to be included in modprobe.conf
/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/opasswd	Contains the password history for check of reuse of old passwords
/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	Contains the counter for the number of consecutive failed login attempts
/var/log/lastlog	Stores time and date of last successful for each user.

Database	Purpose
/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.
/etc/audit/audit.conf	central audit configuration file
/etc/audit/filter.conf	configuration file defining the audit filter rules
/etc/audit/filesets.conf	defines directories and files where access to is required to be audited.
/etc/stunnel/*.conf	Stunnel configuration file
/etc/stunnel/stunnel.pem	File with certificate and private key for Stunnel

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.

6.2.8.6 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.

6.2.8.7 Testing the TOE Protection Mechanisms (TP.7)

The TOE provides a tool for the system administrator that allows him to test the correct functions of the protection features of the underlying abstract machine. This tool performs tests on

- the main memory (to check for failures in the memory hardware) (TP7.1)
- the processor (to check the functions of the memory management unit and the separation between user and kernel mode) (TP7.2)
- I/O devices (to check for correct operation of some I/O devices including the hard disks and the firmware used to access the disks) (TP7.3)

The tool generates a report on the tests performed and the results that those test had. The report is generated in human readable format and may be stored in a file or directed to a printer (TP7.4).

This function contributes to satisfy the security requirement FPT_AMT.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 EAL4 augmented by ALC_FLR.3 are met by SLES 9.

Table 6-5: Mapping Assurance Requirements to Documentation

Assurance Component	Documentation describing how the requirements are met
ACM_AUT.1	Configuration management procedures within SUSE are highly automated using the SUSE Autobuild tool. The functions and usage of this tool are documented in the SUSE Autobuild documentation.
ACM_CAP.4	Configuration management procedures within SUSE are highly automated using a process supported by the AutoBuild tool.
ACM_SCP.2	Source code, generated binaries, documentation, test plan, test cases and test results are maintained under configuration management.
ADO_DEL.2	SLES is delivered on CD / DVD in shrink-wrapped package to the customer. SUSE verifies the integrity of the production CDs / DVDs by checking a production sample. The certification-sles-ibm-eal4.rpm package as well as other packages that contain fixes must be downloaded from the SUSE web site. Since those packages are digitally signed the user is able and has to verify the integrity and authenticity of those packages. Every user can verify the integrity of the packages of the distribution by verifying their digital signature.
ADO_IGS.1	Guidance for installation and system configuration is provided in the Evaluated Configuration Guide [SECGUIDE].
ADV_FSP.2	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. Differences between the man pages (which describe the glibc interface) and the actual system call entry points are covered by a separate document.
ADV_HLD.2	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_IMP.1	As an Open Source product the TOE is delivered with its full source code.
ADV_LLD.1	A document desribing the low-level design of the TSF is provided. This document contains detailed descriptions of the modules that make up the TSF, their interfaces and interrelationships. Also the commonly used data structures are explained. In addition call graphs show the flow and interdependencies between the modules.
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.
ADV_SPM.1	An informal security policy model of the security functions of the TOE is provided as a separate document.
AGD_ADM.1	The Evaluated Configuration Guide [SECGUIDE] and the SLES admin

Assurance Component	Documentation describing how the requirements are met
	handbook plus a special README file contain the specifics for the secure administration of the evaluated configuration.
AGD_USR.1	The Evaluated Configuration Guide [SECGUIDE] and the SLES user handbook plus a special README file contain the specifics for the secure usage of the evaluated configuration.
ALC_DVS.1	The SUSE security procedures are defined and described in documents in the SUSE intranet.
ALC_FLR.3	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.
ALC_LCD.1	The Open Source life cycle model with the activities performed by SUSE as the distributor is provided, showing how the critical aspects of software development and maintenance are covered within this model.
ALC_TAT.1	The tools used for development and testing are defined in a separate document that also explains, how the tools are used within the development and maintenance process.
ATE_COV.2	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_DPT.1	Testing at internal interfaces is defined and described in the test plan documents and the test case descriptions
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_MSU.2	A Misuse Analysis is provided by the sponsor.
AVA_SOF.1	The Strength of Function Analysis has been provided for the mechanism based on permutational or probabilistic algorithms as part of the developer's vulnerability analysis document.
AVA_VLA.2	A vulnerability analysis has been provided that describes the sponsor's approach to identify vulnerabilities of SLES the techniques he used for the vulnerability analysis, the input he used as well as the results of the findings.

6.5 TOE Security Functions requiring a Strength of Function

The TOE has the password based security function for identification and authentication (IA) that is implemented by a probabilistic or permutational mechanism. The TOE also uses random numbers as input to the key generation process for cryptographic keys. The TOE uses prime numbers to generate RSA key pairs. The mechanisms rated in the strength of function analysis are the password mechanism for user authentication, the random number generation mechanism and the algoithm used for primaility testing of numbers. The strength claimed for this functions is SOF-medium. The random number generation is a supporting function used within the Secure Communication security function. The claim applies to this supporting function only, not to the whole of the function Secure Communication.

7 Protection Profile Claims

7.1 PP Reference

This Security Target claims conformance with the „Controlled Access Protection Profile" (CAPP) Version 1.d, 8 October 1999. This Protection Profile was developed by the „Information System Security Organisation" of the National Security Agency of the United States of America.

This Protection Profile is listed on the TPEP web site of NSA as a "Certified Protection Profile".

This Protection Profile was also used as the basis of the evaluation of AIX 5.2, Sun Solaris 8 and HP-UX 11.11.

7.2 PP Tailoring

There is one additional security functional requirement (FMT_SMF.1) that has been added to those defined in the CAPP. The reason is AIS 32, Final Interpretation 065, where the new family FMT_SMF is defined and dependencies from FMT_MSA.1 and FMT_MTD.1 to the new component FMT_SMF.1 have been added. To resolve those new dependencies, FMT_SMF.1 has been added as a security functional requirement in addition to those defined in the CAPP.

Two SFRs (FIA_UAU.1 and FIA_UID.1) defined in the PP have been substituted by hierarchical superior ones (FIA_UAU.2 and FIA_UID.2). This does not affect the compliance to the Protection Profile. Since those components don't imply additional dependencies, the dependency analysis performed on the Protection Profile still applies.

Other requirements (FCS_CKM.1, FCS_CKM.2, FCS_COP.1, FDP_UCT.1, FDP_UIT.1, FMT_MSA.2 and FTP_ITC.1) represent TOE specific extensions to the requirements defined by [CAPP].

Security Functional Requirements have been refined where required by the Protection Profile.

One security functional requirement ("Note 1") is included in [CAPP] as an extension to the requirements defined in part 2 of the Common Criteria. Aspects of conformance of structure and content of Note 1 with the Common Criteria requirements for extensions to part 2 are addressed in the evaluation of the Protection Profile. They are therefore not discussed in this Security Target.

Threats have been added (the Protection Profile only defines policies). Two assumptions on the TOE environment (A.NET_COMP) have been added to reflect the distributed nature of the TOE and the need for entropy used for seeding the random number generator as required by AIS20.

One security objective for the TOE (O.COMPROT) has been added to reflect the objective of being able to establish an Inter-TSF trusted channel between the TOE and another trusted IT product.

The following security objectives for the TOE environment have been added:

OE.ADMIN

OE.MAINTENANCE

OE.SOFTWARE_IN

OE.PROTECT

OE.INFO_PROTECT

OE.RECOVER

OE.SERIAL_LOGIN

OE.HW_SEP

Those objectives are required to cover the specific threats addressing the TOE environment. All objectives are related to physical and procedural security measures and therefore address the TOE non-IT environment.

In addition the Security Target has added security requirements for the IT environment (the processor used) to define the requirement for the underlying processor to provide the functions to implement effective separation of the TSF from untrusted software. This includes the requirements FDP_ACC.1, FDP_ACF.1 and FMT_MSA.3 for the IT environment.

The assurance requirements of the Protection Profile are those defined in the Evaluation Assurance Level EAL3 of the Common Criteria. This Security Target specifies an Evaluation Assurance Level EAL 4 augmented by ALC_FLR.3. Since the Evaluation Assurance Levels in the Common Criteria define a hierarchy, all assurance requirements of the Protection Profile are included in this Security Target. ALC_FLR.3 which has been added to the assurance requirements defined in the CAPP has no dependency on any other security functional requirement or security assurance requirement and is therefore an augmentation that has no effect on the security functional requirements or security assurance requirements stated in the Protection Profile.

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
O.AUDITING	P.ACCOUNTABILITY
O.COMPROT	T.COMPROT, 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	A.MANAGE, 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
T.COMPROT	O.COMPROT
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, O.COMPROT, OE.SOFTWARE_IN
P.ACCOUNTABILITY	O.AUDITING

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.

T.COMPROT: The threat of user data being compromised or modified without being detected is removed by O.COMPROT requiring the ability to set up an Inter-TSF trusted channel between the TOE and another trusted IT product that protects user data being transferred over this channel from disclosure and undetected modification.

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 thereby ensuring that the system has a secure initial 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: Physical 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 willfully 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 objective 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 mechanisms 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.AUTHORIZATION and 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 and O.COMPROT protecting user data during transmission to another trusted IT product.. 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.

P.ACCOUNTABILITY: The policy to provide a means to hold users accountable for their activities is implemented by O.AUDITING providing the TOE with such functionality.

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. Functional requirement “Note 1” has been taken from the Controlled Access Protection Profile [CAPP] and the justification for this extension has been addressed in the evaluation of this protection profile.

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:

Audit

The requirements for auditing have been completely derived from [CAPP]. The rationale for those requirements is:

FAU_GEN.1 defines the events that the TOE is required to be able to audit. Those events are related to the other security functional requirements showing which event contributes to make users accountable for their actions with respect to the requirement. FAU_GEN.2 requires that the events are associated with the identity of the user that caused the event. Of course this can only be done if the user is known (which may not be the case for failed login attempts).

FAU_SAR.1 ensures that authorized administrators are able to evaluate the audit records, while FAU_SAR.2 requires that no other users can read the audit records (since they may contain sensitive information). Taking into account that the amount of audit records gathered may be very large, FAU_SAR.3 requires that the TOE provides the ability to search the audit records for a set that satisfies defined attributes.

To avoid that always all possible audit records are generated (which would result in an unacceptable overhead to the system performance and might easily fill up the available disk space) the TOE is required in FAU_SEL.1 to provide the possibility to restrict the events to be audited based on a set of defined attributes.

Requirement FAU_STG.1 defines that audit records need to be protected from unauthorized deletion and modification to ensure their completeness and correctness. Requirement FAU_STG.3 addresses the aspect that the system detects a shortage in the disk space that can be used to store the audit trail. In this case the administrator is informed about the potential problem and can take the necessary precautions to avoid a critical situation.

FAU_STG.4 addresses the problem that the TOE might not be able to record further audit records (e. g. due to the shortage of some resources). Also in this case the TOE needs to ensure that such a situation can not be misused by a user to bypass the auditing of critical activities. Otherwise a user might deliberately bring the TOE into a situation where it is no longer able to audit critical events just to avoid that a critical action he performs is audited.

Management of audit is addressed by FMT_MTD.1 for both the audit trail and audited events.

Secure Communication

The TOE provides two protocols that allow applications or users to securely communicate with other trusted IT products (which may be other instantiations of the TOE). Those protocols use cryptographic functions to ensure the confidentiality and integrity of the user data during transmission as required by FDP_UCT.1 (confidentiality) and FDP_UIT.1 (integrity). The two protocols – although based on the same library of cryptographic functions – use different cryptographic algorithms to provide the required protection.

Both protocols provide the ability to establish an Inter-TSF trusted channel, as required by FTP_ITC.1.

The secure generation of cryptographic passwords used for secure communications is addressed by FMT_MSA.2.

Discretionary Access Control

FDP_ACC.1 requires the existence of a Discretionary Access Control Policy for file system objects 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 reuse (as required by FDP_RIP.2 and Note 1) is a supporting function that prohibits easy access to information via residuals left in objects when they are re-allocated to another subject or object. 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”. Management of the audit subsystem is expressed by the requirements for the management of the audit trail (FMT_MTD.1 “Management of the Audit Trail”) and the management of the audit events (FMT_MTD.1 “Management of the Audit Events”). Audit trail management is supported by the requirements for the audit review (FAU_SAR.1, FAU_SAR.2 and FAU_SAR.3) as well as the requirements for the protection of the audit trail (FAU_STG.1, FAU_STG.3 and FAU_STG.4). Management of the audit events is supported by the ability to select the events to be audited (FAU_SEL.1). In addition the TOE supports two roles (administrative user and normal user) which is expressed by FMT_SMR.1

Security management also comprises the management of a reliable time stamps. Such time stamps are essential for correct time information within audit records. Times stamps are addressed by FPT_STM.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 TOE also needs to provide a tool that allows the administrator to check the integrity of the underlying hardware. Such ability is addressed by FPT_AMT.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

Objective	Security Functional Requirement
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)

Objective	Security Functional Requirement
	Revocation of Object Attributes (FMT_REV.1)
O.RESIDUAL_INFO	Object Residual Information Protection (FDP_RIP.2) Subject Residual Information Protection (Note 1)
O.MANAGE	Management of Object Security Attributes (FMT_MSA.1) Static Attribute Initialization (FMT_MSA.3) Management of the Audit Trail (FMT_MTD.1) Management of Audited Events (FMT_MTD.1) 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) Abstract Machine Testing (FPT_AMT.1)
O.AUDITING	Audit Data Generation (FAU_GEN.1) User Identity Association (FAU_GEN.2) Audit Review (FAU_SAR.1) Restricted Audit Review (FAU_SAR.2) Selectable Audit Review (FAU_SAR.3) Selective Audit (FAU_SEL.1) Guarantees of Audit Data Availability (FAU_STG.1) Action in Case of Possible Audit Data Loss (FAU_STG.3) Protection of Audit Data Loss (FAU_STG.4) Management of the Audit Trail (FMT_MTD.1) Management of Audited Events (FMT_MTD.1) Reliable Time Stamps (FPT_STM.1)
O.COMPROT	Cryptographic Key Generation (FCS_CKM.1 (1-3)) Cryptographic Key Distribution (FCS_CKM.2 (1-4)) Cryptographic Operation (FCS_COP.1 (1-3)) Basic data exchange confidentiality (FDP_UCT.1) Data Exchange Integrity (FDP_UIT.1) Secure Security Attributes (FMT_MSA.2) Inter-TSF Trusted Channel (FTP_ITC.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.AUDITING

The events to be audited must be defined [FAU_GEN.1], and must be associated with the identity of the user that caused the event [FAU_GEN.2]. An authorized administrator must be able to read the audit records [FAU_SAR.1], but other users must not be able to read audit information [FAU_SAR.2]. The administrative user must be able to search the audit events in the audit trail using defined criteria [FAU_SAR.3] and also must be able to define the events that are audited and the conditions under which they are audited [FAU_SEL.1]. All audit records must be provided with a reliable time stamp [FPT_STM.1]. The audit system must ensure that audit records are not deleted or modified [FAU_STG.1] and are not lost because of shortage of resources [FAU_STG.3 and FAU_STG.4]. The administrative user must be able to manage the audit trail [FMT_MTD.1 “Management of the audit trail”] and the audit events [FMT_MTD.1 “Management of the audit events”].

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] and before a resource is given to a subject [Note 1].

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 the audit subsystem [FMT_MTD.1 “Management of the Audit Trail” and FMT_MTD.1 “Management of the Audit Events”] 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.

The TSF must provide the administrator with tools that allow checking the integrity of the underlying hardware [FPT_AMT.1].

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 enforced.

O.COMPROT

The TSF must be able to establish an Inter-TSF trusted channel between itself and another trusted IT product [FTP_ITC.1] protecting the user data transferred from disclosure [FDP_UCT.1] and undetected modification [FDP_UIT.1]. This TSF uses cryptographic functions in the implementation that require securely generating keys [FCS_CKM.1], distributing keys [FCS_CKM.2] and performing the required cryptographic operations on the user data [FCS_COP.1]. Keys used must be secure enough such that they can not be guessed [FMT_MSA.2]

No security functions for the non-IT environment have been 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 the rationale for the selections and instantiations made in the security requirements section for the security requirements taken from part 2 of the Common Criteria. A rationale is given only for those requirements where selections and instantiations in addition to the ones defined in [CAPP] are provided. For the selections and instantiations performed in [CAPP] to the rationale provided there is referred.

In FAU_GEN.1 the different events that the TOE is able to audit are defined with respect to the SFR they belong to. This list has been taken from [CAPP] and extended with the names of the events and with the SFR that are additional to the ones required by [CAPP].

In FAU_SAR.1 it is expressed that an authorized administrator is able to read all the audit data from the audit log and therefore is able to evaluate the information of the audit trail.

In FAU_SAR.3 it is expressed that an authorized administrator is able to search the audit trail for events matching defined selection criteria where the selection can be performed based on the list of attributes defined in the SFR.

In FAU_STG.1 the requirement for preventing unauthorized modifications of the audit records is expressed.

In FAU_STG.3 the requirement for timely notification of the authorized administrator about a potential shortage in the disk space for the audit trail is expressed, allowing the administrator to take the appropriate measures to overcome the situation before it gets critical.

FCS_CKM.1 has multiple instantiations to reflect the requirements for the generation of symmetric and asymmetric keys to be used by the SSH and SSL protocols to set up and maintain a trusted channel between the TOE and another trusted IT product.

FCS_CKM.2 has multiple instantiations to reflect the different ways for public key exchange, session key exchange and Diffie-Hellman key agreement.

FCS_COP.1 has multiple instantiations to define the different cryptographic algorithms used within the SSL and SSH protocol (with the cipher suites configured for the TOE, which are a subset of the cipher suites allowed in the standards defining those protocols).

In FDP_ACC.1 the different objects that SLES 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 FDP_UCT.1 the requirement for the ability to protect user data from disclosure when being transferred and received is expressed.

In FDP_UIT.1 the requirement for the ability to protect user data from unauthorized modification and insertion when being transferred and received is expressed.

In FIA_ATD.1 nothing has 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, filesystem and effective user ID is expressed. While the filesystem and 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 login ID is maintained and allow tracing activities to the real user that originated them.

In FMT_MSA.1 the ability of the authorized administrator and the owner to modify access rights for objects is expressed. In addition the special role of the owner in the case of IPC objects is expressed.

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 four instantiations of FMT_MTD.1 for

- Audit trail management
- Audit event management
- 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.

FPT_AMT.1 expresses the ability of the authorized administrator to perform the tests of the underlying abstract machine on his demand.

In FTP_ITC.1 the ability to set up a trusted channel between the TOE and another trusted IT product is expressed where either the TOE or the other trusted IT product is allowed to initiate the communication over the trusted channel.

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
FAU_GEN.1	O.AUDITING
FAU_GEN.1	O.AUDITING
FAU_SAR.1	O.AUDITING
FAU_SAR.2	O.AUDITING
FAU_SEL.1	O.AUDITING
FAU_STG.1	O.AUDITING
FAU_STG.3	O.AUDITING
FAU_STG.4	O.AUDITING
FCS_CKM.1(1)	O.COMPROT
FCS_CKM.1(2)	O.COMPROT
FCS_CKM.1(3)	O.COMPROT
FCS_CKM.2(1)	O.COMPROT
FCS_CKM.2(2)	O.COMPROT
FCS_CKM.2(3)	O.COMPROT
FCS_CKM.2(4)	O.COMPROT
FCS_COP.1(1)	O.COMPROT
FCS_COP.1(2)	O.COMPROT
FCS_COP.1(3)	O.COMPROT
FDP_ACC.1	O.DISCRETIONARY_ACCESS
FDP_ACF.1	O.DISCRETIONARY_ACCESS
FDP_RIP.2	O.RESIDUAL_INFO
Note 1	O.RESIDUAL_INFO
FDP_UCT.1	O.COMPROT
FDP_UIT.1	O.COMPROT
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.2	O.COMPROT
FMT_MSA.3	O.DISCRETIONARY_ACCESS, O.MANAGE
FMT_MTD.1 Audit Trail	O.AUDITING, O.MANAGE
FMT_MTD.1 Audited Events	O.AUDITING, O.MANAGE
FMT_MTD.1 User Attributes	O.MANAGE
FMT_MTD.1 Authentication Data	O.AUTHORIZATION, O.MANAGE
FMT_REV.1 User Attributes	O.DISCRETIONARY_ACCESS, O.MANAGE
FMT_REV.1	O.DISCRETIONARY_ACCESS, O.MANAGE

SFR	Objectives
Object Attributes	
FMT_SMF.1	O.MANAGE
FMT_SMR.1	O.MANAGE
FPT_AMT.1	O.ENFORCEMENT
FPT_RVM.1	O.ENFORCEMENT
FPT_SEP.1	O.ENFORCEMENT
FPT_STM.1	O.AUDITING
FTP_ITC.1	O.COMPROT

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 additional 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 performs those initialization 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
FAU_GEN.1	FPT_STM.1 Reliable time stamps	Yes
FAU_GEN.2	FAU_GEN.1 Audit data generation FIA_UID.1 Timing of identification	Yes
FAU_SAR.1	FAU_GEN.1 Audit data generation	Yes
FAU_SAR.2	FAU_SAR.1 Audit review	Yes
FAU_SAR.3	FAU_SAR.1 Audit review	Yes
FAU_SEL.1	FAU_GEN.1 Audit data generation FMT_MTD.1 Management of TSF data	Yes
FAU_STG.1	FAU_GEN.1 Audit data generation	Yes
FAU_STG.3	FAU_STG.1 Protected audit trail storage	Yes
FAU_STG.4	FAU_STG.1 Protected audit trail storage	Yes
FCS_CKM.1	[FCS_CKM.2 Cryptographic key distribution or FCS_COP.1 Cryptographic operation] FCS_CKM.4 Cryptographic key destruction FMT_MSA.2 Secure security attributes	No (see comment below)
FCS_CKM.2	[FDP_ITC.1 Import of user data without security attributes or FCS_CKM.1 Cryptographic key generation] FCS_CKM.4 Cryptographic key destruction FMT_MSA.2 Secure security attributes	No (see comment below)
FCS_COP.1	[FDP_ITC.1 Import of user data without security attributes or FCS_CKM.1 Cryptographic key generation] FCS_CKM.4 Cryptographic key destruction FMT_MSA.2 Secure security attributes	No (see comment below)
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
Note 1	No dependencies	Yes
FDP_UCT.1	[FTP_ITC.1 Inter-TSF trusted channel, or FTP_TRP.1 Trusted path] [FDP_ACC.1 Subset access control, or FDP_IFC.1 Subset information flow control]	yes (FTP_ITC.1 and FDP_ACC.1)
FDP_UIT.1	[FDP_ACC.1 Subset access control, or FDP_IFC.1 Subset information flow control] [FTP_ITC.1 Inter-TSF trusted channel, or FTP_TRP.1 Trusted path]	yes (FTP_ITC.1 and FDP_ACC.1)
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

Security Functional Requirement	Dependencies	Resolved
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 Audit Trail	FMT_SMR.1 Security Roles	Yes
FMT_MTD.1 Audit Events	FMT_SMR.1 Security Roles	Yes
FMT_MTD.1 User Attributes	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_AMT.1	No dependencies	Yes
FPT_RVM.1	No dependencies	Yes
FPT_SEP.1	No dependencies	Yes
FPT_STM.1	No dependencies	Yes
FTP_ITC.1	No dependencies	Yes

Comment

The security functional requirements FCS_CKM.1, FCS_CKM.2 and FCS_COP.1 all have a dependency on FCS_CKM.4 (Cryptographic key destruction). The TOE does not explicitly implement a key destruction function.

Key destruction is performed implicitly for the symmetric session keys used by the Object Reuse function, which ensures that memory used to temporarily store the symmetric session key is cleared before it is assigned to another subject or object. This applies for both main memory as well as disk space (the session keys might be written to disk space as part of the paging function of the TOE. They are not stored in ordinary files).

With respect to the long-term public-private key pairs, the key destruction is performed by deleting the file containing the key. The Object Reuse function of the TOE ensures that the disk space previously allocated to the file storing those keys is cleared before it is assigned to another subject or object.

The other dependencies of those security functional requirements are satisfied. The TOE does not import keys but generates all keys themselves as expressed in the security functional requirement FCS_CKM.1

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 EAL4 has been defined such that no unresolved dependencies exist. The additional assurance component ALC_FLR.3 has no dependencies and therefore there are no unresolved dependencies for assurance components.

8.2.6 Strength of function

This Security Target claims a SOF rating SOF-medium. 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-medium 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.

The TOE uses random numbers as input to the key generation process. The SOF analysis includes a statistical analysis of the distribution of those random numbers as well as an analysis of the entropy of the input used for random number generation. It also includes an analysis of the prime number testing mechsnisms used for primality tests of prime numbers used cryptographic keys. This claim applies to FMT_MSA.2, which is included in this ST as a dependency from FCS_CKM.1, FCS_CKM.2 and FCS_COP.1. The strength of function claim does **not** apply to those requirements for cryptographic functions but only to the supporting function FMT_MSA.2.

No strength of function analysis is performed for the cryptographic algorithms supported by the TOE or for the analysis of cryptographically weak keys.

8.2.7 Evaluation Assurance Level

This security target claims EAL4 augmented with ALC_FLR.3, which is seen appropriate for a controlled environment where attackers only have a low attack potential.

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)
FAU_GEN.1	The audit events are generally defined in **AU** explaining, how the events are generated by the TOE. The System Administrator is able to define the events to be audited, which is described in **SM**
FAU_GEN.2	The concept of a Login ID“ that is kept for a user after his initial login is explained in **AU**. This allows tracing events to the user that caused them even if the user changes his real and / or effective and filesystem user ID (e. g. with the su command or with the execution of a suid program).
FAU_SAR.1	The ability of the authorized administrator to read the audit trail and to convert the audit records into human readable format is explained in **AU**.
FAU_SAR.2	The ability to restrict access to the audit trail to authorized users is addressed in **AU** and enforcement is realized by **DA**.
FAU_SAR.3	The ability of the authorized administrator to search the audit trail for events matching defined search criteria is expressed in **AU**.
FAU_SEL.1	The ability of the authorized administrator to define the events to be audited using predicates and logical expressions is described in **AU** and **SM**.
FAU_STG.1	The use of the TOE’s discretionary access control policy to protect the audit trail and the audit configuration files from access by anybody else than an authorized administrator is defined in **AU**.

SFR	Security Functions (TOE Summary Specification)
FAU_STG.3	The ability to generate a syslog message when the disk space for auditing gets below a limit defined in the audit configuration file is described in **AU**.
FAU_STG.4	The ability to stop processes trying to generate audit records in case the audit trail is full is described in **AU**.
FCS_CKM.1	The multiple instantiations of this security functional requirement are described in **SC** where the SSH v2 and SSL v3 protocols and the cipher suites supported by the evaluated configuration are defined together with the key generation functions used.
FCS_CKM.2	The multiple instantiations of this security functional requirement are described in **SC** where the SSH v2 and SSL v3 protocols and the cipher suites supported by the evaluated configuration are defined together with the key exchange / key negotiation functions used.
FCS_COP.1	The multiple instantiations of this security functional requirement are described in **SC** where the SSH v2 and SSL v3 protocols and the cipher suites supported by the evaluated configuration are defined with the cryptographic algorithms used by the cipher suites.
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, queuing system objects and miscellaneous objects.
Note 1	The object reuse performed before an object is re-assigned to another subject are described in **OR**.
FDP_UCT.1	The description how the confidentiality of user data is protected when using the SSH v2 or SSL v3 protocol is described in **SC**.
FDP_UIT.1	The description how the user data is protected from unauthorized modifications and insertions when using the SSH v2 or SSL v3 protocol is described in **SC**.
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 function 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 initiated 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 initiated 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 objects).
FMT_MSA.2	The acceptance of only secure values is related to the use of secure cryptographic keys. The key generation aspects are discussed in **SC** for

SFR	Security Functions (TOE Summary Specification)
	the different cryptographic algorithms used.
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 Audit Trail	The protection and management of the audit trail is described in **AU** as well as in **SM**. There tools available for converting the audit data to human readable format as well as the tool for searching the audit trail data are described.
FMT_MTD.1 Audited Events	The way an authorized administrator can select the events to be audited is defined in **AU** and **SM**.
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 Authentication 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 Object 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: Audit trail management: **AU** and **SM**. Audit event management: **AU** and **SM**. 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_AMT.1	The ability of the authorized administrator to test the functions of the underlying abstract machine are described in **TP**.
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**.
FPT_STM.1	The function for the generation of a reliable time stamp is defined in **SM**.
FTP_ITC.1	The function for setting up a trusted channel between the TOE and another trusted IT product using the SSH v2 or SSL v3 protocol is described in **SC**.

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. A security domain is enforced by restricting access to security relevant objects to authorized users as stated in FPT_SEP.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 differs only slightly, where those differences are explained in FMT_MSA.1 and FMT_REV.1.

Audit of events is performed to be able to hold users accountable for their activities. Generation of audit records including the login ID of the user is addressed by FAU_GEN.1 and FAU_GEN.2. The availability of the audit trail is addressed by FAU_STG.1, FAU_STG.3, and FAU_STG.4. The audit trail must be secured from unauthorized access as described in FAU_SAR.2. Review of the audit trail by the administrator is discussed in FAU_SAR.1 and FAU_SAR.3. The management of both the audit trail and the audited events is described in FMT_MTD.1 and FAU_SEL.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 reuse makes sense. This is addressed in FDP_RIP.2.

Secure communication is used to protect data in transit between the TOE and trusted IT against disclosure and undetected unauthorized modifications as described in FDP_UCT.1 and FDP_UIT.1. There needs to be a trusted channel between the TOE and other trusted IT as defined in FTP_ITC.1. The generation of cryptographic keys for the mechanisms involved is addressed by FCS_CKM.1; the distribution of such keys is discussed in FCS_CKM.2. The cryptographic algorithms used are detailed in FCS_COP.1. As described in FMT_MSA.2 only secure values are allowed for cryptographic keys.

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 and filesystem user ID and group ID can be changed.

In the management section the requirements for the management User Attributes, Authentication Data, and Audit Configuration 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 included 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 one mechanism in the TSF that is implemented by a permutational or probabilistic mechanism subject to a strength of function analysis within the evaluation of this TOE. For the password based authentication mechanism of the security function IA.1, a minimum strength of SOF-medium is claimed. This is done in accordance with the SOF claim for the related security functional requirement FIA_SOS.1. This claim is consistent with the security objective 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 explicitly excluded from the threat scenario described in section 3.2.

The SOF-medium claim applies also to the random numbers used as input for the process to generate keys for the cryptographic algorithms supported by the TOE. It also applies for the primality tests performed for cryptographic key material that needs to be prime numbers. This is done to support the FTM_MSA.2 security functional requirement included in this ST as a dependency of FCS_CKM.1, FCS_CKM.2 and FCS_COP.1. The analysis is performed to ensure that an attacker with a low or medium attack potential is not able to attack the cryptographic functions using attacks not based on cryptoanalytic methods.

The SOF-medium claim does **not** appliy to the cryptographic algorithms, including the cryptographic properties of the hash functions implemented in the TOE. Excluding cryptographic algorithms and related functions from the strength of function analysis is in compliance with the CEM, remarks on ASE_REQ.1.15, para 424.

The SOF-medium claim applies only to the input provided to the cryptographic function used for the generation of cryptographic keys. No claim about the cryptographic strength of the resulting keys with respect to the algorithms used is made.

Therefore, a strength of SOF-medium is consistent with the description of the TOE environment.

8.4 PP Claims Rationale

The TOE is conformant to the Controlled Access Protection Profile CAPP, as referenced in [CAPP].

One additional security objectives for the TOE (O.COMPROT) has been defined to reflect the ability of the TOE to connect with trusted IT products via trusted channels. Objectives for the TOE environment have been added to this ST in addition to the ones contained in CAPP to allow a more distinguished description of the TOE environment - this does not impact the conformance of this ST to the PP.

All security functional requirements in this ST are inherited from the CAPP and the operations allowed / required by the PP are performed and indicated in bold letters. Two security functional components (FIA_UAU.1 and FIA_UID.1) have been replaced by hierarchical higher ones (FIA_UAU.2 and FIA_UID.2). In both cases the only difference is the fact that no interaction with the TOE is allowed without proper user identification and authentication. This does not modify any of the rationale provided in the PP. In addition FMT_SMF.1 has been added to comply with AIS 32 Final Interpretation 065which defines dependencies of two security functional requirements (FMT_MSA.1 and FMT_MTD.1) included in the PP. To satisfy those requirements the new security functional component has FMT_SMF.1 has been added to the Security Target (anticipating that this security functional requirement will be added in an update to the Controlled Access Protection Profile).

Additional SFRs for the TOE IT environment have been defined to cope with the more distinguished description of the TOE environment - this does not impact the conformance of this ST to the PP.

9 Abbreviations

ACL	Access Control 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
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