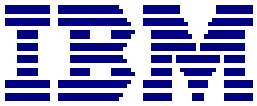

IBM AIX 5L for POWER V5.3 Technology level 5300-05-02 with Innovative Security Systems' PitBull Foundation Suite Release 5.0 and optional IBM Virtual I/O Server Security Target

Version: 1.1.3
Status: Release
Last Update: 2006-11-27

Document History

Version	Date	Changes	Summary	Author
0.1.0	2006-06-21	Entire document	Baseline from PitBull v5.0 for AIX 5.2 ST v1.00 from 2006-03-22.	Scott Chapman, atsec
1.0	2006-06-30	Entire document	Added SED, VIOS, and NFSv4 ACLS.	Scott Chapman, atsec
1.0.1	2006-07-28	Minor	Minor updates to section 2.3, 8.3.2, and table 25 per evaluator comments.	Scott Chapman, atsec
1.1	2006-11-06	Significant	Updated content to reflect the latest CAPP AIX ST	Auston Holt, atsec Scott Chapman, atsec
1.1.1	2006-11-16	Minor	Added IT-env SFR FIA_UID.2. Clarified VIOS text in section 2.2.5.	Scott Chapman, atsec
1.1.2	2006-11-22	Minor	Added POWER5+ to section 2.4.2. Fixed FMT_SMR.1(2) bolding. Removed extraneous TP.10 statements.	Scott Chapman, atsec
1.1.3	2006-11-27	Minor	Added VIOS bullet to section 6.1.5. Fixed typo in section 5.6.10.	Scott Chapman, atsec

Table of Contents

List of Tables

References

[CC] Common Criteria for Information Technology Security Evaluation, August 2005, Version 2.3, Part 1-3, CCIMB-2005-08-001, CCIMB-2005-08-002, CCIMB-2005-08-003.

[CEM] Common Methodology for Information Technology Security Evaluation, August 2005, Version 2.3, CCIMB-2005-08-004.

[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

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

[LSPP] Labeled Security Protection Profile, Version 1.b, 8 October 1999

[PB_AG] PitBull Foundation Suite Administration Guide

[PB_AIXARCH] PitBull AIX 5.3 Security Architecture

1 Introduction

A security target (ST) document provides the basis for the evaluation of an information technology (IT) product or system (e.g., the Target of Evaluation (TOE)). An ST principally defines:

- A set of assumptions about the security aspects of the environment, a list of threats that the product is intended to counter, and any known rules with which the product must comply (Section 3, TOE Security Environment).
- A set of security objectives and a set of security requirements to address security problems (Sections 4 and 5, Security Objectives and IT Security Requirements, respectively).
- The IT security functions provided by the TOE which meet the set of requirements (Section 6, TOE Summary Specification).

The ST for a TOE is a basis for agreement among developers, evaluators, and consumers on the security properties of the TOE and the scope of the evaluation. The audience for a ST may include evaluators, developers and "those responsible for managing, marketing, purchasing, installing, configuring, operating, and using the TOE."

1.1 ST Identification

Title: IBM AIX 5L for POWER V5.3 Technology level 5300-05-02 with Innovative Security Systems' PitBull Foundation Release 5.0 with optional IBM Virtual I/O Server Security Target, Version 1.1.3

Keywords: AIX 5.3, general-purpose operating system, POSIX, UNIX, access control, discretionary access control, information protection, labels, mandatory access control, MLS, Argus Systems Group, security, trusted operating system, PitBull Foundation, VIOS

Publication Date: 2006-11-27

This document is the security target for the CC evaluation of the AIX 5.3 operating system product with optional Virtual I/O Server, and is conformant to the Common Criteria for Information Technology Security Evaluation [CC].

1.2 ST Overview

This security target documents the security characteristics of PitBull Foundation Version 5.0 (PitBull) for the AIX 5.3 operating system.

AIX 5.3 enhanced with PitBull is a highly-configurable UNIX-based operating system which has been developed to meet the requirements of the Labeled Security Protection Profile developed by the Information Systems Security Organization within the National Security Agency to map the TCSEC B1 class of the U.S. Department of Defense (DoD) Trusted Computer System Evaluation Criteria (TCSEC) to the Common Criteria framework. This includes the requirements for Identification and Authentication, Audit, Object Reuse and Access Control including the use of Access Control Lists.

PitBull security enhancements to commercial AIX 5.3 include enhanced Identification & Authentication (I&A), sensitivity labels, authorizations, privileges, labeled printing, network filtering, audit features, Trusted Computing Base (TCB) flags and TCB integrity checking, and kernel security flags. Commercial AIX 5.3 upgraded with Argus security features and assurances provide compatibility with the standard AIX Application Programming Interface (API) and application software.

PitBull enforces mandatory access control (MAC), mandatory integrity control (MIC) and discretionary access control (DAC) policies and can provide network filtering on incoming and outgoing packets. PitBull supports the Internet Protocol (IP) and Common IP Security Option (CIPSO) and provides network filtering based on network interface and host filtering rules.

A summary of PitBull security features can be found in Section 2, TOE Description. A detailed description of PitBull security features can be found in Section 6, TOE Summary Specification. This ST is written with the assumption that the reader has basic knowledge of UNIX.

Several servers running AIX 5.3 enhanced with PitBull can be connected to form a distributed system, but not all components of such a system are components of the TOE. The communication aspects within AIX 5.3 enhanced with PitBull used for this connection are also part of the evaluation. It is assumed that the communication links themselves are protected against interception and manipulation by measures which are outside the scope of this evaluation.

1.3 CC Conformance

This ST is conformant to the Labeled Security Protection Profile, Version 1.b, 8 October 1999 [LSPP].

This ST is CC *Part 2 extended* and *Part 3 conformant*, with a claimed Evaluation Assurance Level of EAL4 augmented by ALC_FLR.1.

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, TOE summary specification and PP claims against [LSPP].

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.

AIX: This documented uses the Term AIX for AIX 5.3H augmented with the PitBull Foundation 5.0 security enhancements.

Administrative User:
This term refers to an administrator of a PitBull Foundation AIX system. Some administrative tasks require the use of authorizations, which can be assigned to one or more user accounts while other tasks can be performed by specified users only.

Authentication data:
This includes a user identifier, password and authorizations for each user of the product.

Object: In AIX, objects belong to one of four categories: file system objects, other kernel objects (such as processes, programs and interprocess communication) and miscellaneous objects. See section 2.2.4 on object reuse for a list of all objects handled by AIX.

Product: The term product is used to define all hardware and software components that comprise the AIX system including VIOS.

Public object: A type of object for which all subjects have read access, but only the TSF or the system administrators have write access.

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

Subject: There are two classes of subjects in AIX:

untrusted internal subject - this is an AIX process running on behalf of some user, running outside of the TSF (for example, with no privileges).

trusted internal subject - this is an AIX 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 AIX product which are connected/networked together and configured to form a usable system.

Target of Evaluation (TOE):
The TOE is defined as the AIX 5.3 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 AIX product. (See below.)

Users can further be categorized as follows:

User	Any entity (human user or external IT entity) outside the TOE that interacts with the TOE.
Human user	Any person who interacts with the TOE.
External IT entity	Any IT product or system, untrusted or trusted, outside of the TOE that interacts with the TOE.
Role	A predefined set of rules establishing the allowed interactions between a user and the TOE.
Identity	A representation (e.g., a string) uniquely identifying an authorized user, which can either be the full or abbreviated name of that user or a pseudonym.
Authentication data	Information used to verify the claimed identity of a user.
Authorized User	A user who may, in accordance with the TSP, perform an operation.

In addition to the above general definitions, this Security Target provides the following specialized definitions:

Access	A right to interact with a system resource.
Advanced Secure Networking (ASN)	The component of the system that labels internal and external network traffic and which mediates access between processes and network resources.
Authorization	An attribute associated with a user account that allows the user to run restricted programs or to run public programs with additional privilege.
Authorized Administrator	A user whose account and session has authorizations allowing privileged, administrative commands to be run.
Category	The non-hierarchical portion of the sensitivity label. The terms ***compartment*** and ***category*** are used interchangeably within this ST.
Classification	The component of a sensitivity label that is hierarchical.
Compartment	See *category*.
Discretionary Access Control (DAC)	A control mechanism that mediates access based on user identity and owner-controlled attributes on objects.
Dominates	Greater than or equal to, as used with labels, privileges, and authorizations.
Integrity Label (TL)	An attribute of a system resource that represents the level of trust associated with the integrity of the resource or data associated with the resource. A TL has only a hierarchical component.
Mandatory Access Control (MAC)	A control mechanism that mediates access based on a label associated with the subject and a label associated with the object and where such labels are not generally under the control of the user/owner associated with the object or subject.
Mandatory Integrity Control (MIC)	A control mechanism that mediates access based on the integrity label associated with the subject and the integrity label associated with the object and where such labels are not generally under the control of the user/owner associated with the object or subject.
Mediation	The act of applying rules to determine access to TOE protected objects.
Privileges	An attribute of a process that allows the process to operate outside of the security policy of the TOE.
Sensitivity Label (SL)	An attribute of system resources that represents the sensitivity of the resource or data associated with the resource. An SL has both a hierarchical and a non-

	hierarchical component.
Trusted Computing Base (TCB)	The software components of the TOE that enforce the TSFs and which must remain inviolate in order to enforce the system security policies.

2 TOE Description

The target of evaluation (TOE) is the operating system AIX Version 5.3 enhanced with PitBull Foundation Version 5.0 and optional IBM Virtual I/O Server (VIOS).

AIX is a general purpose, multi-user, multi-tasking operating system. It is compliant with all major international standards for UNIX systems, such as the POSIX standards, X/Open XPG 4, Spec 1170, and FIPS Pub 180. It provides a platform for a variety of applications in the governmental and commercial environment. AIX is available on a broad range of computer systems from IBM, ranging from departmental servers to multi-processor enterprise servers, and is capable of running in an LPAR (Logical Partitioning) environment.

PitBull Foundation is a security-enhancement product that enforces MAC, MIC, DAC, and TCB control policies to implement security goals, such as confidentiality, integrity, and accountability. PitBull Foundation can operate in a network or stand-alone configuration. In a network configuration, PitBull Foundation supports CIPSO/RIPSO and provides network filtering on incoming and outgoing packets, based on network interface and host filtering rules.

The AIX evaluation shall consist of a closed network of high-end, mid-range and low-end IBM System p5 servers running the TOE.

The TOE Security Functions (TSF) consists of those parts of PitBull Foundation and AIX 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 the system administrator need also to be trusted to manage the system in a secure way. But as with other operating system evaluations they are not considered to be part of this TSF.

a) The hardware and the BootProm firmware are considered to be part of the TOE environment.

b) The TOE shall include installation from CD-ROM and the network.

c) The TOE shall include the following networking applications: Telnet and FTP.

d) The TOE shall include the Virtual Input/Output Server (VIOS) which allows for the virtualization of SCSI drives and network adapters.

System administration tools shall include the *smitty* non-graphical system management tool. The WebSM administrative tool is being excluded.

The TOE environment also includes applications that are not evaluated, but are used as unprivileged tools to access public system services, for example the Mozilla web browser or the Adobe Acrobat Reader to access the supplied online documentation (which is provided in HTML and PDF formats). No HTTP server is included in the evaluated configuration.

2.1 Intended Method of Use

The TOE is a UNIX-based, multi-user, multi-tasking, multilevel secure operating system. The TOE is a multi-user system providing service 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 the system administrator roles.

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 of personal, workgroup, or enterprise computing systems accessed by users local to, or with otherwise protected access to, the computer systems.

The TOE provides facilities for on-line interaction with users. Networking is covered only to the extent to which the TOE can be considered to be part of a centrally-managed system that meets a common set of security requirements.

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

All individual users are assigned a unique user identifier. This user identifier supports individual accountability. 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 other suitably authorized user.

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. In addition, the PitBull component integrated with the TOE allows mandatory access control based on sensitivity labels, integrity controls based on integrity labels and TCB flags, and access control to executable files based on role/authorization.

AIX 5.3 is the platform for a large amount of commercial and scientific applications.

AIX 5.3 complies with the following international standards:

- XPG4 Base 95 Profile (X/Open Portability Guide),
- XPG4 Commands and Utilities V2 (X/Open Portability Guide),
- ANSI/IEEE 1003.2:1992,
- ISO/IEC 9945-2 1993
- FIPS PUB 189 (Effective date April 3, 1995)
- SPEC 1170

AIX 5.3 has significant security extensions compared to standard UNIX systems:

- Access control lists,
- Integrity protection,
- A journaled file system,
- Integrated login framework.

AIX 5.3 provides easy to use interfaces for users and system administrators:

- SMIT (`smitty`) for system and user administration.

PitBull Foundation provides the following additional security mechanisms:

- MAC labels for objects and subjects
- MIC labels for objects and subjects
- User account clearances
- Mandatory access control
- Mandatory integrity control
- TCB integrity protection
- Least privilege
- Multiple administrative and user roles
- Networking controls based on label
- Labeling of printed output

2.2 Summary of Security Features

The following sections present a summary of the security features that the TOE offers.

2.2.1 Identification and Authentication

AIX 5.3 provides identification and authentication based upon user passwords. The quality of the passwords used can be enforced through configuration options controlled by AIX. Other authentication methods (e. g. Kerberos authentication) that are supported by AIX 5.3 in general are not part of the evaluated configuration. Especially pluggable authentication modules that would allow e. g. to use a token based authentication process are not part of the evaluated configuration. The default configuration for authentication is used, which uses passwords to authenticate users. Through the PitBull

Foundation extension, a password generation mechanism is available. PitBull Foundation also provides a mechanism to allow additional site-defined authentication checks.

2.2.2 Auditing

AIX can collect extensive auditing information about security related actions taken or attempted by users, ensuring that users are accountable for their actions.

For each event record, the audit event logger prefixes an audit header to the event-specific information. This header identifies the user and process for which this event is being audited, as well as the time of the event. The code that detects the event supplies the event type and return code or status and optionally, additional event-specific information (the event tail). Event-specific information consists of object names (for example, files refused access or tty used in failed login attempts), subroutine parameters, all security attributes of the subject(s) and object(s) involved in the event, and other modified information.

This audit trail can be analyzed to identify attempts to compromise security and determine the extent of the compromise. The audit tools can also extract audit records of events involving objects and/or subjects having specified security attributes.

2.2.3 Discretionary Access Control

Discretionary Access Control (DAC) restricts access to objects, such as files and is based on Access Control Lists (ACLs) and the standard UNIX permissions for user, group and others. Access control mechanisms also protect IPC objects from unauthorized access. The PitBull Foundation extensions include support for ACLs on network ports and interfaces.

Additionally, VIOS provides DAC between VIOS SCSI device drivers acting on behalf of LPAR partitions as subjects and logical/physical volumes as objects. It also provides DAC between VIOS Ethernet device drivers acting on behalf of groups of LPAR partitions sharing a virtual network and VIOS Ethernet adapter device drivers where one is the subject and the other is the object (the Ethernet packets cannot contain VLAN tags).

2.2.4 Object Reuse

All resources are protected from Object Reuse (*scavenging*) by one of three techniques: explicit initialization, explicit clearing, or storage management. Four general techniques are used to meet this requirement:

- ***Explicit Initialization:*** The resource's contents are explicitly and completely initialized to a known state before the resource is made accessible to a subject after creation.
- ***Explicit Clearing:*** The resource's contents are explicitly cleared to a known state when the resource is returned for re-use.
- ***Storage Management:*** The storage making up the resource is managed to ensure that uninitialized storage is never accessible.
- ***Erase Disk:*** AIX offers as part of its diagnostic subsystem an Erase Disc service aid that can be invoked by the administrator to overwrite all data currently stored in user-accessible blocks of a disk with pre-defined bit patterns.

2.2.5 Security Management

The management of the security critical parameters of AIX is performed by the system administrator. A set of commands that require system administrator 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 the system administrator.

VIOS defines a separate set of roles for system management than AIX. Each VIOS role has a set of commands available to it. Security parameters are stored in specific files that are protected by the access control mechanisms of the TOE against unauthorized access by users.

In addition, PitBull Foundation provides two modes of operation, a configuration mode and an operational mode. The configuration mode is intended to be used for administration of critical, security-relevant parts of the system. The restrictions associated with configuration mode cannot be overridden or bypassed by any mechanism. These restrictions are:

1. all network traffic can be prohibited or new netrules can be loaded
2. the at and cron programs are disabled

2.2.6 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.

TSF software and data, files and directories, kernel objects, IPC and networks sockets/packets are protected by TCB, MAC, MIC, DAC, and process isolation mechanisms.

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

The system administrator has the ability to start a program that checks the hardware for correct operation.

The operational mode of AIX is intended to be the standard operating mode of the machine. The restrictions associated with operational mode cannot be overridden or bypassed by any mechanism. These restrictions are:

1. the kernel security flags cannot be modified
2. FSF_TCB and FSF_TCBPROC flagged objects cannot be created, modified, or deleted

2.2.7 Privileges, Authorizations, Roles, and Superuser Emulation

The TOE implements a privilege mechanism within the kernel that allows users to implement the *least privilege principle*. A privilege is an attribute of a process that allows the process to bypass specific restrictions and limitations of the system. Privileges are associated only with processes, not user accounts. Privileges are used to override security constraints, to permit expanded use of certain system resources such as memory and disk space, and to adjust the performance and priority of the process. Restricting privileges on a process limits the damage that can result if an operation is improperly performed. Untrusted programs must not have any privileges assigned to them.

The TOE least privilege mechanism takes the place of the traditional user ID 0 (superuser/root) mechanism of Unix. In the TOE, user ID 0 is treated exactly like any other system user ID unless superuser emulation is in effect for the process.

Privileges can be associated with executable files and assigned to an executing process, similar to the way the setuid bit on a file modifies the executing process's user ID. A process can also be prevented from acquiring privileges via the exec mechanism. Privileges can be used directly within a user-level program that is responsible for mediating or enforcing security by having the program retrieve its privilege set from the kernel and to make decisions based on the presence or absence of specific privileges. A process can temporarily disable one or more of its privileges if the process needs to perform an action on the system without bypassing the system security policy.

The TOE supports the policy of separation of duties, which provides for the compartmentalization of responsibility reducing the potential damage from a corrupt user or administrator, and places limits on the authority of the user or administrator. Authorizations provide a mechanism to grant rights to users to perform particular actions and run particular programs, such as programs that will run with privileges to bypass MAC, MIC, or DAC limitations. Each authorization has a well-defined set of functions that can be performed by users who are granted that authorization. A user is assigned to one or more authorizations by associating that user's name with the authorization(s). There are two types of authorized users: administrative role users and ordinary users. A user with an administrative role is allowed to use commands that manage the operations of the system.

The concept of administrative roles is implemented in the TOE. Roles are implemented by giving user accounts the appropriate authorizations. Authorizations can be used to implement a four-eyes principle, for example giving one user the authorization to add a user to the system, but not to modify or set the initial password, for which another authorization assigned to another user is needed.

A program has the ability to query the active authorizations associated with the user running the program, and the program can behave differently and use different privileges based on the authorization set of the user running the program. For the evaluated configuration, administrators (or, administrative users) are defined as all users that have any authorization assigned to them, with the exception of the LOGIN authorization that needs to be assigned to any user

who needs to log into the system. All user IDs below 128 are considered system (administrative) IDs, they are typically used for daemons and other trusted applications.

The TOE provides a superuser emulation mechanism that allows the system to operate similar to a standard UNIX system. Superuser emulation can be enabled for specific processes while leaving all other processes running under the standard TOE least privilege and authorization mechanisms. There are five ways in which a process can emulate superuser:

1) A process can be granted all privileges on the system, regardless of its user ID.
2) Using the PV_SU_EMUL privilege, a process can be granted all privileges associated with standard AIX/Unix superuser regardless of its user ID, such as the privileges to bypass any DAC restrictions and to management the auditing mechanism, but not privileges that are specific to the TOE-provided augmentation of standard AIX/UNIX security functionality, such as the privileges to modify kernel authorization tables, override MAC checks, etc.
3) Alternatively, the SU_EMUL system security flag can be set to grant processes all privileges associated with standard AIX/Unix superuser when their process user ID is 0. This mode is disabled in the evaluated configuration.
4) A process can be granted all authorizations/roles regardless of its user ID.
5) A process can be granted a "virtual user ID" of 0 so that queries to the kernel for its user ID will return 0 even regardless of the actual user ID associated with the process.

2.2.8 Mandatory Access Control

PitBull Foundation provides full mandatory access control (MAC) for all objects on the system. Every file, directory, IPC object, and process on the system is given a sensitivity label (SL) which cannot be modified by an unprivileged process. Each user account is assigned a range of valid SLs, and the user can only operate on the TOE within that range. A process (or user) can only create objects at its current SL, and can only read and write objects subject to the MAC restrictions imposed by the system. It is not possible for unauthorized users to "downgrade" information or to bypass MAC restrictions by any utility or application on the system. Copies of a file, or portions of a file, created by any possible means, will always be protected at an SL at least as high as the original file.

2.2.9 Mandatory Integrity Control

PitBull Foundation provides full mandatory integrity control (MIC) for all objects on the system. Every file, directory, IPC object, and process on the system is given an integrity label (TL) which cannot be modified by an unprivileged process. Each user account is assigned a range of valid TLs, and the user can only operate on the TOE within that range. A process (or user) can only create objects at its current TL, and can only read and write objects subject to the MIC restrictions imposed by the system. It is not possible for unauthorized users to "upgrade" integrity levels associated with data or to bypass MIC restrictions by any utility of application on the system. Copies of a file, or portions of a file, created by any possible means, will always be protected at a TL no greater than that of the original file.

2.2.10 TCB Protection

The TOE provides the concept of a Trusted Computing Base (TCB). Kernel, device drivers, system administration utilities, and other critical software that is used to enforce and administer the security of the system is part of this TCB. In addition, any file system object in the TOE (file, directory, device, etc.) can be marked with a TCB flag: FSF_TCB. Alternatively, executables can be marked with the FSF_TCBPROC flag. The TCB is subject to several bypass control mechanisms enforced by the TOE, such as additional access control and integrity protection. Changes to objects being flagged as TCB objects can only be made when the system is in configuration mode and only by a process having the PV_TCB privilege.

The integrity of objects in the TCB database are verified at every system startup and at the request of an authorized administrator.

2.2.11 Advanced Secure Networking (ASN)

The TOE provides export and import of labeled data via network interfaces and enforces mandatory access control for network traffic by means of ASN. ASN provides two sets of networking rules: network interface and host filtering. Both

types of networking rules determine what processing occurs on a packet before its transmission or when it is received. These rules apply sensitivity labels to packets and enforce MAC restrictions on packets according to those labels.

Network interface rules enforce packet label processing based on the physical network interface of the host. Host rules enforce packet label processing based on the source and destination IP addresses (with network masking allowed) of the packet, the source and destination ports of the request, and the protocol being used. Both types of rules provide several criteria for determining which packets to drop and which to pass.

2.3 Software

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

- IBM AIX 5L for POWER V5.3, Program Number 5765-G03, with Recommended Technology Package 5300-05-02 (5.3H) and Innovative Security Systems' PitBull Foundation Suite Release 5.0.
- The Virtual I/O Server (VIOS) contained in IBM Advanced Power Virtualization Version 1.3, Program Number 5765-G30.
- The TOE documentation is supplied on CD-ROM.
- The following table contains a list of LPPs / File Sets that make up the TOE. For each of these "LPP Names" there may be multiple actual installable components with that prefix.

Table 1:List of LPPs / File Sets

LPP Name	Description
bos	AIX Base Operating System
devices	AIX supported devices
printers	AIX printer drivers and control files
sysmgt	System management tools.
argus.bos	PitBull-enhanced AIX bos components
argus.b1mls	PitBull utilities and libraries
argus.integrity	PitBull integrity databases
argus.pitbull	PitBull Foundation man pages

2.4 Configurations

The evaluated configurations are defined as follows.

- The system must be installed according to the PitBull Foundation installation guide [PB_IG].
- AIX 5.3 supports the use of IPv4 and IPv6, but only IPv4 is included in this evaluation.
- Only 64 bit architectures are included.
- Web Based Systems Management (WebSM) is not included.
- Both network (NIM, Network Install Manager) and CD installations are supported.
- Only the default mechanisms for identification and authentication are included. 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 connect directly to the workstation and afforded the same physical protection as the workstation.
- AIX 5.3 provides both a native and a Sys5 print system. Only Sys5 is supported in the evaluated configuration, as it implements the labeling requirements from LSPP.
- System security flags (or, kernel security flags) need to be configured as identified in section 6.2.12.1).
- The system must be configured to disable remote access for an individual user after five consecutively failed login attempts have occurred for this user.

The TOE comprises one of the server machines (and optional peripherals) listed in section 2.4.2 running the system software listed in Table 1 (a server running the above listed software is referred to as a "TOE server" below).

If the product is configured with more than one TOE server, they are linked by LANs, which may be joined by bridges/routers or by TOE workstations which act as routers/gateways or they connect using the Virtual Input/Output Server (VIOS).

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:

- the AIX journaled filesystem *jfs2*,
- the High Sierra filesystem for CD-ROM drives, *cdrfs*.
  - the DVD-ROM file system, *udfs*.
- 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 MAC, MIC, and DAC attributes inferred from the underlying process's and user security attributes.
- The Network File System, *nfs* (V3, V4).

cdrfs, udfs, procfs and (client-side) nfs are single level file systems: For mandatory access control, the labels of their mount point apply to all objects in the mounted file system. Single level file systems are not subject to mandatory integrity control and TCB policies, and their objects cannot be associated with privileges.

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:

- The TOE is running in an LPAR on an IBM System p5 POWER5 and POWER5+ servers.

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

- all terminals supported by the TOE
- all storage devices and backup devices supported by the TOE (hard disks, CDROM drives, streamer drives, floppy disk drives)[1]
- all printer devices supported by the TOE
- Network connectors supported by the TOE (e.g., Ethernet) supporting TCP/IP services over the TCP/IP protocol stack.

[1] The system distinguishes between storage and backup devices. Storage devices are hardware devices holding parts of the AIX file system, such as hard disks and CD ROMs. Backup devices are devices used for archiving data like floppy disks and streamer tapes that do not have a file system. Note that the distinction depends on the actual usage.

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 employed.

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 for 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 inadvertent or casual attempts to breach the system security. The TOE is not intended to be applicable to circumstances in which protection is required against determined attempts by hostile and well funded attackers to breach system security.

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

3.2.1 Threats countered by the TOE

T.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.UAACTION	An undetected violation of the security policy may be caused as a result of an attacker (possibly, but not necessarily, an unauthorized user of the TOE) attempting to perform actions that the individual is not authorized to do.
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 A that tries to impersonate as another authorized user without knowing the authentication information.
T.VIOS	A VIOS SCSI device driver acting on behalf of an LPAR partition may try to access logical volumes or physical volumes that are not assigned to device driver. A VIOS Ethernet device driver acting on behalf of a group of LPAR partitions may try to access a VIOS Ethernet adapter device driver intended for a different VIOS Ethernet device driver (or vise versa).

3.2.2 Threats to be countered by measures within the TOE environment

The following threats are to be countered by the TOE environment.

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 accidentally 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 accidentally without malicious intent.
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 is capable of causing hardware malfunction. Note that such a hardware malfunction may be caused accidentally without malicious intent by persons having physical access to the TOE.
TE.LPAR	When running in a logical partition, software running in a different partition than the TOE is able to access resources that are assigned to the TOE (i.e.,resource that belong to the partition the TOE is running in).

3.3 Organizational Security Policies

The TOE complies with the following organizational security policies:

P.ACCOUNTABILITY	The users of the system shall be held accountable for their actions within the system.
P.AUTHORIZED_USERS	Only those users who have been authorized to access the information within the system may access the system.
P.DATAFLOW	Access to information is based on sensitivity, as represented by a label, associated with that information, and the formal sensitivity clearance of users, to access that information. The access rules enforced prevent a user from accessing information that is of higher sensitivity than the user's sensitivity clearance allows. Furthermore, unauthorized users can not downgrade information to a lower sensitivity.
P.ERASE	Administrators shall be able to support information compartmentalization by preventing recovery of logically deleted information from physically and logically intact SCSI hard disk drives before they are re-used within the same system. Such hard disk drives will remain within the physical and logical protection domain of the TOE and will reside within the TSC.
P.INTEGRITY	The TOE shall be capable of distinguishing between levels of trustworthiness in terms of integrity, and the TOE shall prevent data from being modified by users operating at a lower level of trust.
P.NEED_TO_KNOW	The right to access specific data objects is determined on the basis of: • the owner of the object; and • the identity of the subject attempting the access; and

	• the implicit and explicit access rights to the object granted to the subject by the object owner.
P.STATIC	Dynamic partitioning must not be used for the allocation and de-allocation of resources to the TOE's partition during operation of the TOE. Only "static" partitioning may be performed while the TOE is in a non-operating phase.
P.TCBINTEGRITY	The TOE shall be able to ensure that components of the TCB shall be modified only by authorized administrators. The TOE shall be able to check the integrity of the TCB.
P.DIST_USERS	When the TOE is used in a distributed environment, the administrators shall ensure that the user databases on each TOE are consistent with each other.

3.4 Assumptions

This section indicates the minimum physical and procedural measures required to maintain security of the TOE.

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.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.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_ADM	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.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.CONNECT	All connections to peripheral devices and all network connections reside within the controlled access facilities. Internal communication paths to access points such as terminals or other systems are assumed to be protected against unauthorized access and the loss of integrity and confidentiality of the information transmitted.
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.

3.4.4 Procedural Aspects

A. CLEARANCE	Procedures exist for granting users authorization for access to specific security levels.

A. SENSITIVITY	Procedures exist for establishing the security level of all information imported into the system, for establishing the security level for all peripheral devices (e.g., printers, tape drives, disk drives) attached to the TOE, and marking a sensitivity label on all output generated.

4 Security Objectives

4.1 Security Objectives for the TOE

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.AUTHORIZATION	The TOE must ensure that only authorized users gain access to the TOE and its resources.
O.DISCRETIONARY_ACCESS	The TSF must control accessed to resources based on identity of users. The TSF must allow authorized users to specify which resources may be accessed by which users.
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.ERASE	The TOE shall offer a mechanism to overwrite user-accessible blocks of SCSI hard disk drives with pre-defined bit patterns.
O.MANAGE	The TSF must provide all the functions and facilities necessary to support the authorized administrators that are responsible for the management of TOE security and must ensure that only authorized administrators are able to access such functionality.
O.MANDATORY_ACCESS	The TOE must control access to resources based upon the sensitivity and categories of the information being accessed and the sensitivity clearance of the subject attempting to access that information.
O.MANDATORY_INTEGRITY	The TOE must control access to resources based on the integrity level of the information being accessed and the integrity level of the subject attempting to access that information.
O.NETWORK_ACCESS	The TOE must control access between the TOE and other systems based on host security attributes and the network interface on which packets are sent or received.
O.RESIDUAL_INFORMATION	The TOE must ensure that any information contained in a protected resource is not released when the resource is recycled.
O.TCB_ACCESS	The TOE must control write access to and provide integrity checks for resources protected as part of the trusted computing base as specified by an authorized administrator.
O.STACK	The TOE shall offer a mechanism to prevent the execution of code on the stack of selected processes.
O.VIOS	The TSF must control access between LPAR partitions and logical/physical volumes and VIOS SCSI device drivers acting on behalf of a group of LPAR partitions. The TSF must allow authorized users to specify which VIOS SCSI device drivers. The TSF must control access between VIOS Ethernet adapter device drivers and VIOS Ethernet device drivers acting on behalf of groups of LPAR partitions sharing a virtual network. The TSF must allow authorized users to specify which VIOS Ethernet adapter device drivers may be accessed by which VIOS Ethernet device driver acting on behalf of a group of LPAR partitions sharing a virtual network.

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 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 sensitivity clearance of the system. The media on which authentication data is stored must not be physically removable from the system by unauthorized users. Users must not disclose their passwords to other individuals.
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.
OE.INFO_PROTECT	Those responsible for the TOE must establish and implement procedures to ensure that information is protected in an appropriate manner. In particular: • All network and peripheral cabling must be approved for the transmittal of the most sensitive data held by the system. Such physical links are assumed to be adequately protected against threats to the confidentiality and integrity of the data transmitted. • DAC protections on security critical files (such as audit trails and authentication databases) shall always be set up correctly. • User sensitivity clearance and MAC protection must always be set up correctly by specifying appropriate security levels for all information imported into the system, peripheral devices and output generated and granting users authorization to access only specific security levels.
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.MAINTENANCE	Administrators of the TOE must ensure that the comprehensive diagnostics facilities provided by the product are invoked at every scheduled preventative maintenance period.
OE.PHYSICAL	Those responsible for the TOE must ensure that those parts of the TOE critical to security policy are protected from physical attack which might compromise IT security objectives.
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.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.SOFTWARE_IN	Those responsible for the TOE shall ensure that the system shall be configured so that only an administrator can introduce new trusted software into the system.

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

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

The following security objective applies when the TOE is running on underlying machines that have more than one logical partition configured:

OE.LPAR	The underlying hardware must protect the resources assigned to the logical partition the TOE is running in against access from software running in a different logical partition.

5 Security Requirements

Selection and assignment operations on IT security requirements have been marked **bold**. Refinements are set underlined. Iterations are identified by additional identifiers in parentheses as part of the component reference. All substitutions of security functional requirements to LSPP and all additional security functional requirements beyond LSPP are described in section 7.2.

Application notes from LSPP have been taken over unaltered into this ST to allow easy comprehension of the Protection Profile's intention and its implementation for AIX with PitBull and have been marked as such. Additional application notes have been added as necessary. Please note that application notes are by nature informative and supposed to aid the reader's comprehension of the TSP and their implementation by the TOE.

5.1 Extended Component Definitions

5.1.1 FDP_RIP.3-AIX

The Security Target defines an extended component FDP_RIP.3-AIX as part of the FDP_RIP family in CC Part 2 for usage within this ST.

Component leveling

FDP_RIP.3-AIX Hard disk drive residual information protection requires that the TSF ensure that any residual information content of a hard disk drive that is being formatted is made unavailable for logical recovery ("erased") upon administrator-invoked de-allocation, or formatting, of the hard disk drive.

Management: FDP_RIP.3-AIX

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

a. The choice of when to erase a hard disk drive, and which hard disk drive, should be made configurable within in the TOE.

The choice of bit patterns used to overwrite the blocks of the hard disk drive, and how often to overwrite the blocks, could be made configurable within the TOE.

Audit: FDP_RIP.3-AIX

There are no events identified that should be auditable if FAU_GEN Security audit data generation is included in the ST.

FDP_RIP.3-AIX Hard disk drive residual information protection

Hierarchical to: No other components.

FDP_RIP.3-AIX.1	The TSF shall overwrite all data stored in currently user-accessible blocks of a hard disk drive with pre-defined bit patterns upon request of the authorized administrator.

Dependencies: No dependencies.

5.1.2 FPT_RVM.2-AIX

The Security Target defines an extended component FPT_RVM.2-AIX as part of the FPT_RVM family in CC Part 2 for usage within this ST.

Component leveling

FPT_RVM.2-AIX Stack execution reference mediation requires that the TSF ensure that code on the stack of selected processes cannot be executed on the stack.

Management: FPT_RVM.2-AIX

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

a. The choice of which processes are monitored and which are not monitored should be made configurable within the TOE.

Audit: FPT_RVM.2-AIX

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

a. Minimal: disabling of the monitoring of one or more processes.

FPT_RVM.2-AIX Stack execution reference mediation

Hierarchical to: No other components.

Dependencies: No dependencies.

FPT_RVM.2-AIX.1 **The TSF shall provide the ability to allow/deny the execution of code residing on the stack of a process created from the executable to anyone who can write to the executable and allow an authorized administrator to override this setting.**

5.2 TOE Security Functional Requirements

The following is a table identifying the SFRs for the TOE used in this ST, their origin, and the performed operations.

Table 2: SFRs for the TOE

SFR	Origin	Component
FAU_GEN.1	LSPP	AIX
FAU_GEN.2	LSPP	AIX
FAU_SAR.1	LSPP	AIX
FAU_SAR.2	LSPP	AIX
FAU_SAR.3	LSPP	AIX
FAU_SEL.1	LSPP	AIX
FAU_STG.1	LSPP	AIX
FAU_STG.3	LSPP	AIX
FAU_STG.4	LSPP	AIX
FDP_ACC.1(1)	LSPP	AIX
FDP_ACC.1(2)	CC Part 2	AIX
FDP_ACC.1(3)	CC Part 2	AIX
FDP_ACC.1(4)	CC Part 2	VIOS
FDP_ACF.1(1)	LSPP	AIX
FDP_ACF.1(2)	CC Part 2	AIX
FDP_ACF.1(3)	CC Part 2	AIX
FDP_ACF.1(4)	CC Part 2	VIOS
FDP_ETC.1	LSPP	AIX
FDP_ETC.2	LSPP	AIX
FDP_IFC.1(1)	LSPP	AIX
FDP_IFC.1(2)	CC Part 2	AIX
FDP_IFC.1(3)	CC Part 2	AIX
FDP_IFF.2(1)	LSPP	AIX
FDP_IFF.2(2)	CC Part 2	AIX
FDP_IFF.2(3)	CC Part 2	AIX
FDP_ITC.1	LSPP	AIX
FDP_ITC.2	LSPP	AIX
FDP_RIP.2	LSPP	AIX
Note 1	LSPP	AIX
FDP_RIP.3-AIX	ECD (section 5.1.1)	AIX
FIA_ATD.1(1)	LSPP	AIX
FIA_ATD.1(2)	CC Part 2	VIOS
FIA_SOS.1	LSPP	AIX, VIOS
FIA_UAU.2	LSPP *	AIX, VIOS
FIA_UAU.7	LSPP	AIX, VIOS

SFR	Origin	Component
FIA_UID.2	LSPP *	AIX, VIOS
FIA_USB.1(1)	LSPP	AIX
FIA_USB.1(2)	CC Part 2	VIOS
FMT_MSA.1(1)	LSPP	AIX
FMT_MSA.1(2)	CC Part 2	AIX
FMT_MSA.1(3)	CC Part 2	AIX
FMT_MSA.1(4)	CC Part 2	AIX
FMT_MSA.1(5)	CC Part 2	AIX
FMT_MSA.1(6)	CC Part 2	VIOS
FMT_MSA.3(1)	LSPP	AIX
FMT_MSA.3(2)	CC Part 2	AIX
FMT_MSA.3(3)	CC Part 2	AIX
FMT_MSA.3(4)	CC Part 2	AIX
FMT_MSA.3(5)	CC Part 2	AIX
FMT_MSA.3(6)	CC Part 2	VIOS
FMT_MTD.1(1)	LSPP	AIX
FMT_MTD.1(2)	LSPP	AIX
FMT_MTD.1(3)	CC Part 2	AIX
FMT_MTD.1(4)	LSPP	AIX, VIOS
FMT_MTD.1(5)	LSPP	AIX, VIOS
FMT_MTD.1(6)	CC Part 2	AIX
FMT_MTD.1(7)	CC Part 2	VIOS
FMT_REV.1(1)	LSPP	AIX
FMT_REV.1(2)	LSPP	AIX
FMT_REV.1(3)	CC part 2	VIOS
FMT_SMF.1	CC Part 2	AIX, VIOS
FMT_SMR.1(1)	LSPP	AIX
FMT_SMR.1(2)	CC Part 2	VIOS
FPT_AMT.1	LSPP	AIX
FPT_RVM.1	LSPP	AIX
FPT_RVM.2-AIX	ECD (section 5.1.2)	AIX
FPT_SEP.1	LSPP	AIX
FPT_STM.1	LSPP	AIX
FPT_TDC.1	CC Part 2	AIX
FPT_TST.1	CC Part 2	AIX

* hierarchically higher components than in LSPP were chosen

5.2.1 Security Audit (FAU)

5.2.1.1 Audit Data Generation (FAU_GEN.1)

The TSF shall be able to generate an audit record of the auditable events listed in column "Event" of Table 3 (Auditable Events). This includes all auditable events for the basic level of audit, except FIA_UID.1's user identity during failures. FAU_GEN.1.1/NOTE 4

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

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

b. The sensitivity labels of subjects, objects, or information involved; and

c. The additional information specified in the "Details" column of Table 3 (Auditable Events)

Application Note from LSPP: 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.

Application Note for AIX: The execution of tests of the underlying machine and the results of the test are not stored in the normal audit trail but in a separate Diagnostic Error Log file. This file is protected by the MAC, MIC, and DAC such that only the System Administrator can access the file. In order to record/add an audit record to the audit log, an process needs the PV_AU_ADD or PV_AU privilege. When an installation requires auditing the use of the diag command, object auditing can be used where the diag command file is the object and execution is the access mode.

Table 3: Auditable Events

Component	Event	Details (Event Names)
FAU_GEN.1	Start-up and shutdown of the audit functions.	**start_up: AUD_It (cmd=1) shutdown: AUD_It (cmd=4)**
FAU_GEN.2	None	
FAU_SAR.1	Reading of information from the audit records.	**See below**[1]
FAU_SAR.2	Unsuccessful attempts to read information from the audit records.	**FILE_Open**
FAU_SAR.3	None	
FAU_SEL.1	All modifications to the audit configuration that occur while the audit collection functions are operating.	**AUD_Bin_Def, AUD_Events, AUD_Objects, AUD_Proc**
FAU_STG.1	None	
FAU_STG.3	Actions taken due to exceeding of a threshold.	**AUD_Lost_Recs**
FAU_STG.4	Actions taken due to the audit storage failure.	**AUD_Lost_Recs**
FDP_ACC.1(1)	None	
FDP_ACC.1(2)	**None**	
FDP_ACC.1(3)	**None**	
FDP_ACC.1(4)	**None**	
FDP_ACF.1(1)	All requests to perform an operation on an object covered by the SFP.	**FILE_Mode** **FILE_Owner** **FILE_Chpriv** **FILE_Facl** **FILE_Fchmod** **FILE_Fchown** **FILE_Fchpriv** **FILE_Link** **FILE_Mknod** **FILE_Open (for create)** **FILE_Rename** **FILE_Truncate** **FILE_Unlink** **FS_Rmdir** **FS_Mount** **FS_Umount** **MSG_Owner** **MSG_Mode** **MSG_Delete** **MSG_Create** **SEM_Owner** **SEM_Create**

[1] Object auditing can be used to define the events to be audited specifically for the audit files. This feature allows to define the whole set of events that should be audited for those files. Object auditing allows to define all actions on the audit file that should generate an audit record.

Component	Event	Details (Event Names)
		SEM_Delete **SEM_Mode** **SHM_Create** **SHM_Delete** **SHM_Owner** **SHM_Mode** **ASG_SetFAuth,** **ASG_LSetFAuth,** **ASG_FSetFAuth**
FDP_ACF.1(2)	**All requests to perform an operation on an object covered by the SFP.**	**ASG_SetPTCBMode**
FDP_ACF.1(3)	**All requests to perform an operation on an object covered by the SFP.**	**ASG_SetFAuth,** **ASG_LSetFAuth,** **ASG_FSetFAuth,** **ASG_SetPAuth**
FDP_ACF.1(4)	**None**	
FDP_ETC.1	All attempts to export information.	**ADT_PRT_JOB,** **ASG_EConnect,** **ASG_ESend**
FDP_ETC.2	All attempts to export information.	**ADT_PRT_JOB,** **ADT_PRT_FILE,** **ASG_PRINT,** **ASG_EConnect,** **ASG_ESend**
FDP_ETC.2	Overriding of human-readable output marking. (Additional)	**ASG_LABCAT**
FDP_IFC.1(1)	None	
FDP_IFC.1(2)	**None**	
FDP_IFC.1(3)	**None**	
FDP_IFF.2(1)	All decisions on requests for information flow.	**RFM_MAC_Pass,** **RFM_MAC_Fail**
FDP_IFF.2(2)	**All decisions on requests for information flow.**	**ASG_ESend**
FDP_IFF.2(3)	**All decisions on requests for information flow.**	**RFM_MAC_Pass,** **RFM_MAC_Fail,** **RFM_TL_Pass,** **RFM_TL_Fail**
FDP_ITC.1	All attempts to import user data, including any security attributes.	**ASG_ERead,** **ASG_EReadv,** **ASG_EReceive**
FDP_ITC.2	All attempts to import user data, including any security attributes.	**ASG_ERead,** **ASG_EReadv**
FDP_RIP.2	None	
Note 1	None	
FDP_RIP.3-AIX	**None**	
FIA_ATD.1(1)	None	
FIA_ATD.1(2)	**None**	
FIA_SOS.1	Rejection or acceptance by the TSF of any tested secret.	**USER_Login** **PASSWORD_Change** **USER_SU**
FIA_UAU.**2**	All use of the authentication mechanism.	**USER_Login** **PASSWORD_Change** **USER_SU**

Component	Event	Details (Event Names)
FIA_UAU.7	None	
FIA_UID.**2**	All use of the user identification mechanism, including the identity provided during successful attempts.	**USER_Login PASSWORD_Change USER_SU**
FIA_USB.1(1)	Success and failure of binding user security attributes to a subject (e.g.,success and failure to create a subject).	**PROC_Execute PROC_RealUID PROC_AuditID PROC_SetUserIDs PROC_RealGID PROC_SetGroups PROC_Environ**
FIA_USB.1(2)	**None**	
FMT_MSA.1(1) **through (5)**	All modifications of the values of security attributes.	**PROC_Environ PROC_Privilege PROC_Execute PROC_RealUID PROC_AuditID PROC_SetUserIDs PROC_RealGID PROC_SetGroups, ASG_SetFAuth, ASG_LSetFAuth, ASG_FSetFAuth, ASG_SetFFSF, ASG_LSetFFSF, ASG_FSetFFSF, ASG_SetFLab, ASG_LSetFLab, ASG_FSetFLab, ASG_SetFPV, ASG_LSetFPV, ASG_FSetFPV, ASG_SetKAuths, ASG_SetKATUids, ASG_SetTLibpath, ASG_SetSecconf, ASG_ASNINIT, ASG_NETRULE**
FMT_MSA.1(6)	**None**	
FMT_MSA.3(1) **through (5)**	Modifications of the default setting of permissive or restrictive rules. All modifications of the initial value of security attributes.	**Administrator-defined object write auditing events for /etc/security/user, /etc/security/limits, /etc/security/clear, /etc/security/las, /etc/asn/rules.host and /etc/asn/rules.int**
FMT_MSA.3(6)	**None**	
FMT_MTD.1(1)	All modifications to the values of TSF data.	**Object write auditing events for the TSF files containing the TSF data**
FMT_MTD.1(2)	All modifications to the values of TSF data.	**PROC_Sysconfig AUD_it AUD_Bin_Def, ASG_ASNINIT,**

Component	Event	Details (Event Names)
		ASG_NETRULE
FMT_MTD.1(3)	**All modifications to the values of TSF data.**	**Object write auditing events for the /etc/security/audit/config file,**
FMT_MTD.1(4)	All modifications to the values of TSF data.	**USER_Change**
FMT_MTD.1(5)	All modifications to the values of TSF data.	**USER_Change PASSWORD_Change PASSWORD_Flags**
FMT_MTD.1(6)	**All modifications to the values of TSF data.**	**ASG_SetFPV, ASG_LSetFPV, ASG_FSetFPV**
FMT_MTD.1(7)	**None**	
FMT_REV.1(1)	All attempts to revoke security attributes.	**USER_Change**
FMT_REV.1(2)	All modifications to the values of TSF data.	**FILE_Acl**
FMT_REV.1(3)	**None**	
FMT_SMF.1	**None.**	
FMT_SMR.1(1)	Modifications to the group of users that are part of a role. Every use of the rights of a role. (Additional / Detailed)	**USER_Change PROC_Privilege, RFM_Priv_Pass, RFM_Priv_Fail, RFM_AZ_Pass, RFM_AZ_Fail**
FMT_SMR.1(2)	None	
FPT_AMT.1	Execution of the tests of the underlying machine and the results of the test.	**Diagnostic Error Log or object auditing (see Application Note)**
FPT_RVM.1	None	
FPT_RVM.2-AIX	**Disabling of the stack execution detection.**	**SEDMGR_File, SEDMGR_Odm**
FPT_SEP.1	None	
FPT_STM.1	Changes to the time.	**PROC_Adjtime**
FPT_TDC.1	**None.**	
FPT_TST.1	**None.**	

5.2.1.2 User Identity Association (FAU_GEN.2)

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

Application Note from LSPP: 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.

Application Note for AIX: AIX stores the identity of the user in the header field “ah_ruid” and “ah_luid” of each audit record. For a description of the difference between the “real user ID” and the “login user ID” see chapter 6.

5.2.1.3 Audit Review (FAU_SAR.1)

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

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

Application Note from LSPP: The minimum information which must be provided is the same that which is required to be recorded in 5.2.1. The intent of this requirement is that there exists a tool for the 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 5.2.5 and 5.2.6. It is expected that a single tool will exist within the TSF which will satisfy all of these requirements.

Application Note for AIX: The access control to audit files within AIX is regulated by the MAC, MIC, and DAC of AIX, as well as the authorization subsystem. Furthermore, in order to read files marked as audit files, a process needs the PV_AU_READ or PV_AU privilege. It is the task of the administrator to ensure that the audit files as well as the audit configuration files are protected appropriately. Tools are provided to the administrator to read and format the audit records. For a more detailed description see chapter 6.

5.2.1.4 Restricted Audit Review (FAU_SAR.2)

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

Application Note from LSPP: 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.

Application Note for AIX: This requirement is satisfied by the access control facility of AIX. It is the task of the system administrator to manage the read access right to the audit files appropriately. See Application Note for FAU_SAR.1.

5.2.1.5 Selectable Audit Review (FAU_SAR.3)

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

a. User identity;

b. Subject sensitivity label;

c. Object sensitivity label;

d. MAC success or failures;

e. MIC success or failures;

f. DAC success or failures;

g. PRIV success or failures;

h. FSF success or failures;

i. AUTH success.

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

Application Note for AIX: AIX provides two commands for audit data processing: *auditpr* and *auditselect*. Details are described in chapter 6.

5.2.1.6 Selective Audit (FAU_SEL.1)

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

a. User identity;

b. Subject sensitivity label;

c. Object sensitivity label;

and the following additional attributes:

a. file name

b. event type

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

Application Note for AIX: The main configuration data for the audit is stored in the file */etc/security/audit/config.* This file together with */etc/security/user* allows the administrator to include or exclude auditable events based on the identity of the user. In addition the audit configuration file */etc/security/audit/objects* allows to include or exclude auditable events based on the file name. In order to modify the audit system configurations of the TOE, a process needs the PV_AUD_ADMIN or PV_AU privilege.

Application Note for AIX: When the file security flag FSF_MONITOR is enabled for a file, the audit subsystem will record accesses to the file regardless of other audit classes in the audit mask of a process.

5.2.1.7 Guarantees of Audit Data Availability (FAU_STG.1)

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

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

Application Note from LSPP: 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.

Application Note from AIX: Protection of stored audit records from deletion and modifications is performed using the access control mechanisms of AIX. In order to allow a process to write or delete a file marked as an audit file, or to mark a file as an audit file, this process needs the PV_AU_WRITE or PV_AU privilege.

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

The TSF shall generate an alarm to the authorized administrator if the file system holding the audit trail falls below **a configurable limit of free blocks**. FAU_STG.3.1 / NOTE 3

Application Note from LSPP: 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.

Application Note from AIX: AIX 5.3 has been modified to implement such a variable and have it configurable.

5.2.1.9 Prevention of Audit Data Loss (FAU_STG.4)

The TSF shall be able to prevent auditable events, except those taken by the authorized administrator, and **either stop the system in panic mode or count the number of audit records lost** if the audit trail is full. FAU_STG.4.1 / NOTE 5

Application Note from LSPP: 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.

Application Note for AIX: In the case all audit bins are full, AIX 5.3 can be configured to stop execution (panic) or count the number of audit records lost. Normal execution can only be resumed after space for the audit bin is available. This has to be achieved by an authorized administrator, that starts the system in single-user mode and perform the necessary actions to make disk space available for auditing.

5.2.2 User Data Protection (FDP)

5.2.2.1 Discretionary Access Control Policy (FDP_ACC.1(1))

The TSF shall enforce the Discretionary Access Control (DAC) Policy on **processes** acting on the behalf of users **as subjects and file system objects (ordinary files, directories, device special files, UNIX Domain socket special files, named pipes), IPC objects (message queues, SysV semaphores, shared memory segments)** and all operations among subjects and objects covered by the DAC policy. FDP_ACC.1.1

Application Note from LSPP: 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. This security requirement does not apply to non-SysV semaphores. 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.

Application Note for AIX: See chapter 6 for details of the Discretionary Access Control capabilities for the different types of subjects.

5.2.2.2 TCB Access Control Policy (FDP_ACC.1(2))

The TSF shall enforce the **TCB Policy** on **processes acting on the behalf of users as subjects and jfs2 file system objects (ordinary files, directories, device special files, UNIX Domain socket special files, named pipes) and all operations among subjects and objects covered by the TCB Policy**. FDP_ACC.1.1

5.2.2.3 Authorization Policy (FDP_ACC.1(3))

The TSF shall enforce the **Authorization Policy** on **processes acting on the behalf of users as subjects, functions implemented in executable files in jfs2 file systems as objects, and the attempts of processes to invoke such functions as operations**. FDP_ACC.1.1

5.2.2.4 VIOS Access Control Policy (FDP_ACC.1(4))

The TSF shall enforce the **VIOS Access Control Policy** on**:** FDP_ACC.1.1

- **Volumes: VIOS SCSI device drivers acting on behalf of LPAR partitions as subjects with Logical Volumes and Physical Volumes as objects and the operations among subjects and objects as covered by the policy**
- **Network: VIOS Ethernet device drivers acting on behalf of a group of LPAR partitions sharing a virtual network and VIOS Ethernet adapter device drivers (where either one can be the subject and the other the object) and the operations among subjects and objects as covered by the policy**.

Application Note for AIX: This requirement applies to VIOS only. This requirement is taken directly from CC part 2, not from the LSPP requirements; thus, the LSPP audit requirements do not apply.

5.2.2.5 Discretionary Access Control Functions (FDP_ACF.1(1))

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

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

b. The following access control attributes associated with an object:

File system objects:

AIXC policy:

permission bits, extended permissions and privilege sets. (Permission bits are the standard UNIX permission bits for user, group, world. Extended permissions can be used to grant or deny access to the granularity of a single user or group using Access Control Lists).

Access rights for file system objects are:
- read

- write
- execute (ordinary files)
- search (directories)

NFSv4 policy:

permission bits or fine grained permissions. (NFSv4 policy where the access rights listed below apply to the following entities: owner, group, everyone. The access rights can be used to allow or deny access to the granularity of a single entity.)

Access rights for system objects are:

- read data (ordinary files)

- list contents (directories)

- write file data (ordinary files)

- add a file (directories)

- append data (ordinary files)

- add subdirectory (directories)

- read extended attributes

- write extended attributes

- execute (ordinary files)

- search (directories)

- delete an object within a directory

- delete the associated object

- read core object attributes (size, time, etc.)

- write core object attributes

- read ACL contents

- write ACL contents

- change ownership (user or group)

- synchronize

IPC objects:

permission bits

Access rights for IPC objects are:
- read
- write

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

File system objects:

AIXC Policy:

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 type of access is within the union of all permission rights (grant entries) defined in the access control list of the object for the subject and is not within the logical union of all restrictions (deny entries) defined in the access control list of the object for the subject. If no entry in the extended permissions either allows or denies access, the access right defined in the permission bits apply. In any other case access is denied.

NFSv4 Policy:

A subject must have search permission for every element of the pathname and the requested access for the object. A subject has the requested type access to an object if all requested access types are specifically allowed before reaching an entry that denies one or more requested types or before reaching the end of the ACL. Otherwise, the requested access is denied.

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 user ID of the current process. Access of a process to an IPC object is allowed, if

the user ID of the of the current process is equal to the user ID of the IPC object creator or owner and the "owner" permission bit for the requested type of access is set or

the 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

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

Privilege sets override DAC decisions in order to allow a process to

1. **Change its GID (PV_DAC_GID);**
2. **Bypass DAC ownership restrictions (PV_DAC_O);**
3. **Bypass DAC ownership restrictions for non-system files only (PV_DAC_O_NS);**
4. **Bypass DAC read restrictions (PV_DAC_R);**
5. **Bypass DAC read restrictions for non-system files only (PV_DAC_R_NS);**
6. **Send a signal to an unrelated process (PV_DAC_SIG);**
7. **Change its UID (PV_DAC_UID);**
8. **Bypass DAC write restrictions (PV_DAC_W);**
9. **Bypass DAC write restrictions for non-system files only (PV_DAC_W_NS);**
10. **Bypass DAC execute/search restrictions (PV_DAC_X);**
11. **Bypass DAC execute/search restrictions for non-system files only (PV_DAC_X_NS);**
12. **obtain all of the privileges listed in 1. to 12. (PV_DAC).**

File System Objects:

Process has the appropriate privileges to perform the specific access request. In addition for the NFSv4 policy, the object owner is always allowed to read/write the ACL contents and read/write the core object attributes.

The TSF shall explicitly deny access of subjects to objects based on **write requests to objects that reside on a file system that is mounted read-only**. FDP_ACF.1.4

Application Note from LSPP: A LSPP 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 5.3.2. 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 5.3.2. 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.

Application Note for AIX: The Discretionary access control mechanism is explained in more detail in chapter 6. The details of the handling of discretionary access control for the different types of objects is explained there

5.2.2.6 TCB Access Control Functions (FDP_ACF.1(2))

The TSF shall enforce the **Trusted Computing Base (TCB) Policy** to objects based on the following: **the tcb_enabled kernel security flag, TCB flag attribute for file system objects and the PV_TCB privilege for processes**. FDP_ACF.1.1

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

a) If kernel security flag tcb_enabled is set to *enabled* and the system is in operational mode

A process may not write to a file with the FSF_TCB flag set.

If the FSF_TCBPROC flag is set for an executable, a corresponding process may only load shared libraries that have the FSF_TCB flag set.

A process may not set or clear the FSF_TCB or FSF_TCPROB flag on an object.

b) If kernel security flag tcb_enabled is set to *enabled* and the system is in maintenance mode

A process with the PV_TCB privilege will ignore the file's FSF_TCB or FSF_TCBPROC flag when attempting to access files.

A process with the PV_TCB privilege may set or clear the FSF_TCB or FSF_TCBPROC flag on an object.

The TSF shall explicitly authorize access of subject to objects based on the following additional rules: **none**. FDP_ACF.1.3

The TSF shall explicitly deny access of subject to objects based on **no additional rules**. FDP_ACF.1.4

Application Note for AIX: As spelled out in section 6.2.12.1, the evaluated configuration mandates that *tcb_enabled* be enabled, therefore no policy for the TCB being disabled is specified here.

5.2.2.7 Authorization Functions (FDP_ACF.1(3))

The TSF shall enforce the **Authorization Policy** to objects based on the following: **the user identity, authorizations as the attributes that are assigned to users, and TOE-defined functions associated with the privilege**. FDP_ACF.1.1

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

If a function within an executable requires a specific authorization, the function will only be executed if the executing user has been assigned the proper authorization.

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

Privilege sets override Authorization decisions in order to allow a process to

1. **Modify the kernel authorization table and to change the LAS associated with a process (PV_AZ_ADMIN);**
2. **Retrieve the entire kernel authorization table (PV_AZ_READ);**
3. **Pass all authorization checks, except as restricted by the process's Limiting Authorization Set (PV_AZ_ROOT);**
4. **Set authorizations on executable files (PV_AZ_SET);**
5. **Be identified as running under UID (PV_AZ_EMUL).**

The TSF shall explicitly deny access of subject to objects based on **no additional rules**. FDP_ACF.1.4

5.2.2.8 VIOS Access Control Functions (FDP_ACF.1(4))

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

- **Volumes: A logical volume or physical volume (object) can only be mapped to (accessed by) one VIOS SCSI device driver acting on behalf of an LPAR partition (subject) and this mapping is the access control rule**
- **Network: A VIOS Ethernet device driver acting on behalf of a group of LPAR partitions sharing a virtual network and a VIOS Ethernet adapter device driver (where either one can be the subject and the other the object) can only be mapped to each other in a one-to-one mapping and this mapping is the access control rule**.

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

- **Volumes: If the logical volume or physical volume is mapped to a VIOS SCSI device driver acting on behalf of an LPAR partition, then the device driver can access the logical volume or physical volume, respectively; otherwise, access is denied**
- **Network: If a VIOS Ethernet device driver acting on behalf of a group of LPAR partitions sharing a virtual network is mapped to a VIOS Ethernet adapter device driver, then the device drivers can exchange untagged packets; otherwise, access is denied**.

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

- **none**

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

Application Note for AIX: This requirement applies to VIOS only. This requirement is taken directly from CC part 2, not from the LSPP requirements; thus, the LSPP audit requirements do not apply. An untagged packet is a packet that does not contain a VLAN (Virtual LAN) identifier/tag.

5.2.2.9 Export of Unlabeled User Data (FDP_ETC.1)

The TSF shall enforce the Mandatory Access Control (MAC) Policy when exporting unlabeled user data, controlled under the MAC policy, outside the TSC. FDP_ETC.1.1

The TSF shall export the unlabeled user data without the user data's associated security attributes. FDP_ETC.1.2

The TSF shall enforce the following rules when unlabeled user data is exported from the TSC: NOTE 6

a) Devices used to export data without security attributes cannot be used to export data with security attributes unless the change in device state is performed manually and is auditable;

b) the subject exporting data on the behalf of a user must have appropriate privileges as specified in FDP_IFF.2(1) and FDP_IFF.2(2).

Application Note from LSPP: An LSPP-conformant TOE must provide protections to data exported outside the control of the TSC via any communications mechanisms that do not provide security attributes along with the actual data. The device, or mechanism, used to export information must, itself, have security attributes that correspond to those of the information being exported. The ability to export information must be allowed under the existing rules that establish the MAC policy of the TOE.

Human readable hard copy output must be properly marked with appropriate labels on the top and bottom of pages and on the banner pages at the beginning and end of each output. The ST author must explicitly state the procedures under which this will be accomplished (e.g., use of pre-labeled paper is allowable).

The ST author must also explicitly state the rules under which authorized users can designate the security attributes of the mechanisms, or devices, used to export data without security attributes. The ST author must also make it clear that mechanisms, or devices, used to export data without security attributes cannot also be used to export data with security attributes. Unless this change in state can only be done manually and is audited.

Single-level Input/Output devices and single-level communication channels are not required to maintain the sensitivity labels of the information they process.

5.2.2.10 Export of Labeled User Data (FDP_ETC.2)

The TSF shall enforce the Mandatory Access Control Policy when exporting labeled user data, controlled under the MAC policy, outside the TSC. FDP_ETC.2.1

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

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

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

a) When data is exported in a human-readable or printable form:

- The authorized administrator shall be able to specify the printable label, which is assigned to the sensitivity label associated with the data.
- Each print job shall be marked at the beginning and end with the printable label assigned to the "least upper bound" sensitivity label of all the data exported in the print job.
- Each page of printed output shall be marked with the printable label assigned to the "least upper bound" sensitivity label of all the data exported to the page. By default this marking shall appear on both the top and bottom of each printed page.

b) Devices used to export data with security attributes cannot be used to export data without security attributes unless the change in device state is performed manually and is auditable;

c) Devices used to export data with security attributes shall completely and unambiguously associate the security attributes with the corresponding data;

d) No additional labeled data export rules.

Application Note from LSPP: The ST author may establish rules that control the export of information from the TSC. These rules must reflect the nature of both the object types and the actual object security attributes. In all cases the TOE must export the security attributes with the corresponding information. An LSPP-conformant TOE must only use protocols to export data with security attributes that provide unambiguous pairings of security attributes and the information being exported. Further, the ST author must make it clear that the mechanisms, or devices, used to export data with security attributes cannot be used to export data without security attributes unless this change in state can only be done manually and is audited. In addition, the security attributes must be exported to the same mechanism or device as the information. Also, any change in the security attributes settings of a device must be audited. Explicit rules must exist in the ST for the export of information that represent hardcopy output. The rules must capture the labeling requirements that must be met for printing labels on the first and last pages, top and bottom of pages, etc.; and any overriding of printed labels must be audited. Further, the ST must make certain that the external form of the security attributes, or label, must accurately and unambiguously represent the internal label.

Application Note for AIX: When data is exported via FTP, the label of the user initiating the transaction is associated with the FTP transmission. When data is exported via a printer, the backend filtering mechanism of the print system implements the required labeling functionality. The printer hardware and software (printer engine) are outside the evaluated TOE boundary.

5.2.2.11 Mandatory Access Control Policy (FDP_IFC.1(1))

The TSF shall enforce the Mandatory Access Control (MAC) Policy on FDP_IFC.1.1

a) the subjects listed in Table 4 acting on the behalf of users;

b) the named objects in Table 4;

and all operations among subjects and objects covered by the MAC policy as defined in Table 4.

Table 4: MAC SFP Subjects, Objects, and Operations

Subject	Object	Named Object	Operations between Subject/Named Object
processes acting on behalf of a specific	FSO	device special files – block and character, TCB	Read/Write/Exec

Subject	Object	Named Object	Operations between Subject/Named Object
user		directory – regular	
		file – regular, system, audit	
		symbolic link	
	IPC	message	Read/Write/Exec
		semaphore	
		shared memory	
	Miscellaneous	signal vector	Read/Write
		STREAMS message block	
		pipe – unnamed (FIFO)	

Application Note from LSPP: 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 ST author must also explicitly list the objects that exist in the TOE. This list must include storage objects. Objects should include data storage resources as well as input/output devices, etc. The operations, listed in the ST, among subjects and objects must explicitly define all relationships between subjects and objects in the TOE, and must be consistent with the list of objects defined in the earlier assignment. A subject is an entity within the TSC that causes operations to be performed.

5.2.2.12 Mandatory Integrity Control Policy (FDP_IFC.1(2))

The TSF shall enforce the **Mandatory Integrity Control (MIC) Policy** on FDP_IFC.1.1

a) the subjects listed in Table 5 acting on the behalf of users;

b) the named objects in Table 5;

and all operations among subjects and objects covered by the MIC policy as defined in Table 5.

Table 5: MIC SFP Subjects, Objects, and Operations

Subject	Object	Named Object	Operations between Subject/Named Object
processes acting on behalf of a specific user	jfs2 objects	device special files – block and character, TCB	Write
		directory – regular	
		file – regular, system, audit	
		symbolic link	

5.2.2.13 Mandatory Advanced Secure Networking Policy (FDP_IFC.1(3))

The TSF shall enforce the **Advanced Secure Networking (ASN) Policy** on FDP_IFC.1.1

a) hosts identified by IP addresses as the subjects;

b) data packets to be transferred between hosts as objects;

and the transfer of data packets from and to hosts via network connections.

5.2.2.14 Mandatory Access Control Functions FDP_IFF.2 (1)

The TSF shall enforce the Mandatory Access Control Policy based on the following types of subject and information security attributes: FDP_IFF.2.1

a) The sensitivity label of the subject; and

b) The sensitivity label of the object containing the information.

Sensitivity label of subjects and objects shall consist of the following:

- A hierarchical level; and
- A set of non-hierarchical categories.

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

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

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

c) If the sensitivity label of subject A is greater than or equal to the sensitivity label of subject B; then the flow of information from subject B to subject A is permitted.

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

The TSF shall provide **no additional MAC capabilities**. FDP_IFF.2.4

The TSF shall explicitly authorize an information flow based on the following rules: FDP_IFF.2.5

1. Privilege sets override MAC decisions in order to bypass

a) sensitivity clearance restrictions (PV_MAC_CL);

b) MAC restrictions when

1. **files flagged as being exempt from MAC (PV_MAC_OVRRD);**
2. **sending a signal (PV_MAC_W_PROC) or when getting information about a process (PV_MAC_R_PROC), provided that the target process's label is within the acting process's sensitivity clearance;**

c) all MAC READ restrictions (PV_MAC_R);

d) MAC READ restrictions when

1. **the object's label is within the process's sensitivity clearance (PV_MAC_R_CL);**
2. **reading a STREAM message block, provided that the message's label is within the process's sensitivity clearance (PV_MAC_R_STR);**

e) all MAC WRITE restrictions (PV_MAC_W);

f) MAC WRITE restrictions when

1. **when the process label is greater than or equal to the object's label and the object's label is within the process's sensitivity clearance (PV_MAC_W_DN);**
2. **when the process label is less then or equal to the object's label and the object's label is within the process's sensitivity clearance (PV_MAC_W_UP);**

g) a combination of all other MAC privileges (PV_MAC).

2. SIGCHILD is exempt from the checks in FDP_IFF.2.2.

3. For partitioned directories, signals, and network streams: if the sensitivity label of the object is greater than the sensitivity label of the subject, then the flow of information from the subject to the object is permitted (write operation).

The TSF shall explicitly deny an information flow based on the following rules: **no additional rules**. FDP_IFF.2.6

The TSF shall enforce the following relationships for any two valid sensitivity labels: FDP_IFF.2.7

a) There exists an ordering function that, given two valid sensitivity labels, determines if the sensitivity labels are equal, if one sensitivity label is greater than the other, or if the sensitivity labels are incomparable, and

- Sensitivity labels are equal if the hierarchical level of both labels are equal and the non-hierarchically category sets are equal.
- Sensitivity label A is greater than sensitivity label B if one of the following conditions exists:

- If the hierarchical level of A is greater than the hierarchical level of B, and the non-hierarchical category set of A is equal to the non-hierarchical category set of B.
- If the hierarchical level of A is equal to the hierarchical level of B, and the non-hierarchical category set of A is a proper super-set of the non-hierarchical category set of B.
- If the hierarchical level of A is greater than the hierarchical level of B, and the non-hierarchical category set of A is a proper super-set of the non-hierarchical category set of B.

- Sensitivity labels are incomparable if they are not equal and neither label is greater than the other.

b) There exists a "least upper bound" in the set of sensitivity labels, such that, given any two valid sensitivity labels, there is a valid sensitivity label that is greater than or equal to the two valid sensitivity labels and

c) There exists a "greatest lower bound" in the set of the sensitivity labels, such that, given any two valid sensitivity labels, there is a valid sensitivity label that is not greater than the two valid sensitivity labels

Application Note from LSPP: The terms "security attribute" and "information flow control security attribute" refer to the sensitivity labels of subjects and objects. A LSPP-conformant TOE should support at least 16 site definable hierarchical levels and 64 site definable non-hierarchical categories. The implementation of sensitivity labels does not need to store labels in a format which has the components of the label explicitly instantiated, but may use some form of tag which maps to a level and category set.

Application Note for AIX: The AIX with PitBull security policy model mandates equality when writing, which has been reflected by refining FDP_IFF.2.2 b).

Application Note for AIX: AIX with PitBull allows to assign label ranges to devices and directories (see also section 6.1.5). Effectively, in such cases the MAC policy is applied to each of the labels in the label range and permits information flow if the subject's label matches any one of the object's labels as required by the policy.

5.2.2.15 Advanced Secure Networking (ASN) Policy (FDP_IFF.2 (2))

The TSF shall enforce the **ASN Policy** based on the following types of subject and information security attributes: FDP_IFF.2.1

a) Subject(s):

1. **IP address;**

b) information security attributes associated with objects:

1. **IP address packet source and destination;**
2. **protocol;**
3. **port (source and destination);**
4. **network interface;**
5. **IPSO labels;**
6. **IPSO security attributes;**
7. **minimum and maximum SL (only when using CIPSO).**

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

a) if the source/destination IP address of the packet is equal to the source/destination IP address specified in the rule;

b) if the IP address of the packet is within the network mask specified for the rule;

c) if the direction of the packet flow corresponds to the direction of the rule (IN/OUT);

d) if the protocol of the packet is equal to the protocol specified in the rule;

e) if the source/destination port is within the source/destination port range specified in the rule;

f) if the network interface of the packet is equal to the network interface specified in the rule;

g) if the IPSO labels are within the range defined by the rule, and rule set to allow IPSO labels;

h) if the packet's SL is within the minimum and maximum SL specified for the rule.

The TSF shall enforce **no additional ASN SFP rules**. FDP_IFF.2.3

The TSF shall provide **no additional ASN SFP capabilities**. FDP_IFF.2.4

The TSF shall explicitly authorize an information flow based on the following rules: FDP_IFF.2.5

Privilege sets override ASN decisions in order to allow a process to

a) set kernel ASN tables (PV_ASN_ADMIN);

b) perform restricted ioctl calls to drivers (PV_ASN_IOCTL)

c) open a restricted port (PV_ASN_PORT);

d) obtain the privileges in a) to c) (PV_ASN).

The TSF shall explicitly deny an information flow based on **no additional rules**. FDP_IFF.2.6

The TSF shall enforce the following relationships for any two valid information flow control security attributes: FDP_IFF.2.7

a) There exists an ordering function that, given two valid security attributes, determines the security attributes are equal, if one security attribute is greater than the other, or if the security attributes are incomparable; and

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

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

Application Note for AIX: RIPSO header are only able to apply hierarchical labels to the data transferred, RIPSO packets do not contain information about any compartments.

5.2.2.16 Mandatory Integrity Control Policy (FDP_IFF.2 (3))

The TSF shall enforce the **Mandatory Integrity Control (MIC) Policy** based on the following types of subject and information security attributes: FDP_IFF.2.1

a) The integrity label (TL) of the object containing the information.

b) Integrity labels of subjects and objects shall consist of a hierarchical level.

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

a) If the integrity label of the subject is greater than or equal to the integrity label of the object; then the flow of information from the subject to the object is permitted (a write operation).

The TSF shall enforce **no additional MIC SFP rules**. FDP_IFF.2.3

The TSF shall provide **no additional MIC SFP capabilities**. FDP_IFF.2.4

The TSF shall explicitly authorize an information flow based on the following rules: FDP_IFF.2.5

Privilege sets override MIC decisions in order to

a) bypass integrity restrictions (PV_MIC).

The TSF shall explicitly deny an information flow based on **no additional rules**. FDP_IFF.2.6

The TSF shall enforce the following relationships for any two valid information flow control security attributes: FDP_IFF.2.7

a) There exists an ordering function that, given two valid security attributes, determines the security attributes are equal, if one security attribute is greater than the other, or if the security attributes are incomparable; and

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

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

Application Note for AIX: Read enforcement for mandatory integrity control is disabled in the evaluated configuration, and consequently no corresponding rule is specified in FDP_IFF.2.2.

5.2.2.17 Import of Unlabeled User Data (FDP_ITC.1)

The TSF shall enforce the Mandatory Access Control Policy when importing unlabeled user data, controlled under the MAC Policy, from outside the TSC. FDP_ITC.1.1

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

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

a) Devices used to import data without security attributes cannot be used to import data with security attributes unless the change in device state is performed manually and is auditable;

b) **Only users with authorization can import unlabeled data from outside the TSC.**

c) Only authorized administrators can specify the default label for imported data.

Application Note from LSPP: The LSPP-conformant TOE must provide protections for data imported from outside the control of the TSC via functions that do not provide reliable security attributes along with the actual data. The imported data must be assigned a sensitivity label that will be used to enforce the MAC policy. Further, the ability for a subject to import information must be controlled under the existing rules that establish the MAC policy of the TOE. The ST author must explicitly state the rules under which authorized users can designate the security attributes of the mechanisms, or devices, used to import data without security attributes; and any attribute change must be audited. The ST author must also make it clear that mechanisms, or devices, used to import data without security attributes cannot also be used to import data with security attributes unless this change in state can only be done manually and is audited.

5.2.2.18 Import of Labeled User Data (FDP_ITC.2)

The TSF shall enforce the Mandatory Access Control Policy when importing labeled user data, controlled under the MAC SFP from outside the TSC. FDP_ITC.2.1

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

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

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

The TSF shall enforce the following rules when importing labeled user data controlled under the MAC SFP from outside the TSC: FDP_ITC.2.5

a) Devices used to import data with security attributes cannot be used to import data without security attributes unless the change in device state is performed manually and is auditable;

b) no additional importation control rules.

c) Sensitivity label, consisting of the following:

- A hierarchical level; and
- A set of non-hierarchical categories.

Application Note from LSPP: The ST author must provide for the protection of data imported from outside the control of the TSC via any mechanisms that provide security attributes along with the information being imported. The security attributes received along with the data must accurately represent the security attributes of the data with which they are associated. The ST author must make it clear that the mechanisms, or devices used to import data with security attributes cannot be used to import data without security attributes unless this change in state can only be done manually and is audited. Also, any change in the security attributes of a device must be audited. A LSPP-conformant TOE should support at least 16 site definable hierarchical levels and 64 site definable non-hierarchical categories. The

implementation of sensitivity labels does not need to store labels in a format which has the components of the label explicitly instantiated, but may use some form of tag which maps to a level and category set.

Rationale from LSPP: This component supports the O.MANDATORY_ACCESS objective by defining the rules which will be enforced by the TOE.

5.2.2.19 Object Residual Information Protection (FDP_RIP.2)

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

Application Note from LSPP: 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 de-allocation from objects is sufficient to satisfy this requirement, if unallocated resources will not accumulate new information until they are allocated again.

Application Note for AIX: Chapter 6 describes for each object type how object reuse is handled.

5.2.2.20 Subject Residual Information Protection (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 from LSPP: 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 de-allocation from subjects is sufficient to satisfy this requirement, if unallocated resources will not accumulate new information until they are allocated again.

Application Note for AIX: This requirement was added in the Labeled Security Protection Profile to address resources that are not directly allocated to objects. Chapter 6 explains in detail how object reuse is handled for many kind of resources, also those that are not objects defined in this Security Target.

Application Note for AIX: Chapter 6 describes how residual information protection for processes is handled.

5.2.2.21 Hard disk drive residual information protection (FDP_RIP.3-AIX)

The TSF shall overwrite all data stored in currently user-accessible blocks of a hard disk drive with pre-defined bit patterns upon request of the authorized administrator. FDP_RIP.3-AIX.1

Application note for AIX: This requirement applies to SCSI drives only. Chapter 6 describes further how residual information protection for SCSI hard disk drives is implemented.

5.2.3 Identification and Authentication (FIA)

5.2.3.1 User Attribute Definition (FIA_ATD.1(1))

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

a) User Identifier;

b) Group Memberships;

c) Authentication Data;

d) User Sensitivity and Integrity Clearances;

e) Security-relevant Roles (User Authorizations); and

f) **Default SL;**

g) Default TL;

h) Audit Classes;

i) Password Aging Data

Application Note from LSPP: 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.

Application Note for AIX: The roles that are referenced here are the PitBull relevant roles.

5.2.3.2 User Attribute Definition (FIA_ATD.1(2))

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

a) User identifier;

b) Group Memberships;

c) Authentication Data;

d) Security-relevant Roles.

Application Note for AIX: The roles that are referenced here are the VIOS relevant roles.

5.2.3.3 Strength of Authentication Data (FIA_SOS.1)

The TSF shall provide a mechanism to verify that secrets meet the following: FIA_SOS.1

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 from LSPP: The method of authentication is unspecified by the LSPP, but must be specified in a ST. The method which is used must be shown to have low probability that authentication data can be forged 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 passwords to dictionary attacks. The strength of whatever mechanism implemented must be subjected to a strength of function analysis. (See 6.7.2)

Application Note for AIX: The TOE supports a number of configuration parameters that allow a system administrator to define a specific password policy. With a well-defined password policy and a clear guideline for users how to select passwords that are hard to guess the requirement can be satisfied. Additionally, the TOE supports the ability to restrict the number of failed attempts. The claimed strength of function for this mechanism is: SOF-medium.

5.2.3.4 Authentication (FIA_UAU.2)

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

Application Note from LSPP: 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 of possible user in gaining access to the TOE. This could include help facilities or the ability to send a message to authorized administrators.

Application Note for AIX: AIX does not allow any TSF mediated action of a user that is not authenticated.

5.2.3.5 Protected Authentication Feedback (FIA_UAU.7)

The TSF shall provide only obscured feedback to the user while the authentication is in progress. FIA_UAU.7

Application Note from LSPP: 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.

5.2.3.6 Identification (FIA_UID.2)

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

Application Note from LSPP: 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.

Application Note for AIX: AIX does not allow any TSF mediated action of a user that is not identified. The Labeled Security Protection Profile specifies FIA_UID.1 which allows the definition of actions that a user may perform before being identified. Since FIA_UID.2 is hierarchical to FIA_UID.1, conformance to the Protection Profile is achieved.

5.2.3.7 User-Subject Binding (FIA_USB.1(1))

The TSF shall associate the following user security attributes with subjects acting on the behalf of that user: FIA_USB.1.1 / NOTE 2

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) The sensitivity label used to enforce the MAC SFP, which consists of the following:

- A hierarchical level; and
- A set of non-hierarchical categories.

e) The integrity label used to enforce the MIC SFP;

f) User sensitivity clearance;

g) User integrity clearance;

h) Security relevant roles (user authorizations);

i) Privilege sets and privilege authorization sets associated with the subject being activated;

j) Audit Classes.

The TSF shall enforce the following rules on the initial association of user security attributes with subjects acting on the behalf of a user: FIA_USB.1.2 / NOTE 2

a) The sensitivity label associated with a subject shall be within the sensitivity clearance range of the user;

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

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

The TSF shall enforce the following rules governing changes to the user security attributes associated with subjects acting on the behalf of a user: FIA_USB.1.3 / NOTE 2

a) The effective user ID of a user can be changed by the use of an executable with the setuid bit set. In this case the program is executed with the effective user ID of the program owner. Access rights are then evaluated using the effective user ID of the program owner. The login user ID is not changed with this process, so all audit records can be traced to the real user that executes the program.

b) The effective user ID of a user can be changed by the su command. In this case the effective user ID of the user is changed to the user specified in the su command (provided authentication is successful). The login user ID remains unchanged, so all audit records can be traced to the real user that executes the program.

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

evaluated using the effective group ID of the program owner. The login user ID is not changed with this process, so all audit records can be traced to the real user that executes the program.

Application Note from LSPP: 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 5.4.6, 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.

Application Note for AIX: While privilege sets and privilege authorization sets are actually associated with subjects (i.e.,executables being activated on the behalf of a user as processes) rather than being user security attributes in the narrower sense, they have been added to this requirement to emphasize the fact that they become part of the subject's security attributes relevant for its execution.

5.2.3.8 User-subject Binding (FIA_USB.1(2))

The TSF shall associate the following user security attributes with subjects acting on the behalf of that user: FIA_USB.1.1

a) User identity;

b) Group memberships;

c) Security-relevant roles.

The TSF shall enforce the following rules on the initial association of user security attributes with subjects acting on the behalf of users: FIA_USB.1.2

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

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

The TSF shall enforce the following rules governing changes to the user security attributes associated with subjects acting on the behalf of users: FIA_USB.1.3

a) The effective userID 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 userID of the program owner. Access rights are then evaluated using the effective userID of the program owner. The login userID is not changed with this process.

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

c) The effective groupID of a user can be changed by the use of an executable with the setgid bit set. In this case the program is executed with the effective groupID of the program owning group. Access rights are then evaluated using the effective groupID of the program owner. The login userID is not changed with this process.

5.2.4 Security Management (FMT)

5.2.4.1 Management of Object Security Attributes (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 **authorized administrators and the owner of the object. In the case of objects with NFSv4 ACLs, the system administrator and the owner can modify the access control attributes, plus other users can be granted permission within the ACL to modify the access control attributes of the object.** FMT_MSA.1.1

The TSF shall enforce the Mandatory Access Control Policy to restrict the ability to modify the sensitivity label associated with an object to **authorized administrators**. FMT_MSA.1.1

Application Note from LSPP: 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.

5.2.4.2 Management of Object Security Attributes (FMT_MSA.1 (2))

The TSF shall enforce the **ASN SFP** to restrict the ability to **modify** the security attributes **information flow control attributes representing the ASN rules** to **authorized administrators**. FMT_MSA.1.1

5.2.4.3 Management of Object Security Attributes (FMT_MSA.1 (3))

The TSF shall enforce the **TCB SFP** to restrict the ability to **change_default** the security attributes **FSF_TCB and FSF_TCBPROC** to **authorized administrators**. FMT_MSA.1.1

5.2.4.4 Management of Object Security Attributes (FMT_MSA.1 (4))

The TSF shall enforce the **MIC SFP** to restrict the ability to **modify** the security attributes **integrity labels** to **authorized administrators**. FMT_MSA.1.1

5.2.4.5 Management of Object Security Attributes (FMT_MSA.1 (5))

The TSF shall enforce the **Authorization SFP** to restrict the ability to **modify** the security attributes **authorizations** to **authorized administrators**. FMT_MSA.1.1

5.2.4.6 Management of Object Security Attributes (FMT_MSA.1 (6))

The TSF shall enforce the **VIOS Access Control Policy** to restrict the ability to **modify** the security attributes**:** FMT_MSA.1.1

- **For Volumes: mapping SCSI device drivers acting on behalf of LPAR partitions to logical volumes and physical volumes**
- **For Network: mapping of Ethernet device drivers acting on behalf of a group of LPAR partitions sharing a virtual network to Ethernet adapter device drivers**

to **system administrators only**.

Application Note for AIX: This requirement applies to VIOS only. This requirement is taken directly from CC part 2, not from the LSPP requirements; thus, the LSPP audit requirements do not apply.

5.2.4.7 Static Attribute Initialization (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.3.1

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

The TSF shall allow the **authorized administrators and the owner of the object for Discretionary Access Control and authorized administrators for MAC** to specify alternative initial values to override the default values when an object or information is created. FMT_MSA.3.2

Application Note from LSPP: A LSPP-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.

5.2.4.8 Static Attribute Initialization (FMT_MSA.3 (2))

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

The TSF shall allow the **authorized administrators** to specify alternative initial values to override the default values when an object or information is created. FMT_MSA.3.2

5.2.4.9 Static Attribute Initialization (FMT_MSA.3 (3))

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

The TSF shall allow the **authorized administrators** to specify alternative initial values to override the default values when an object or information is created. FMT_MSA.3.2

5.2.4.10 Static Attribute Initialization (FMT_MSA.3 (4))

The TSF shall enforce the **MIC SFP** to provide **restrictive** default values for security attributes that are used to enforce the **MIC SFP**. FMT_MSA.3.1

The TSF shall allow the **authorized administrators** to specify alternative initial values to override the default values when an object or information is created. FMT_MSA.3.2

5.2.4.11 Static Attribute Initialization (FMT_MSA.3 (5))

The TSF shall enforce the **Authorization SFP** to provide **restrictive** default values for security attributes that are used to enforce the **Authorization SFP**. FMT_MSA.3.1

The TSF shall allow the **authorized administrators** to specify alternative initial values to override the default values when an object or information is created. FMT_MSA.3.2

5.2.4.12 Static Attribute Initialization (FMT_MSA.3 (6))

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

The TSF shall allow **no one** to specify alternative initial values to override the default values when an object or information is created. FMT_MSA.3.2

Application Note for AIX: This requirement applies to VIOS only. This requirement is taken directly from CC part 2, not from the LSPP requirements; thus, the LSPP audit requirements do not apply.

5.2.4.13 Management of the Audit Trail (FMT_MTD.1 (1))

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

Application Note from LSPP: 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.

5.2.4.14 Management of Audited Events (FMT_MTD.1 (2))

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

Application Note from LSPP: 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.

Application Note for AIX: DAC/MAC is used to protect the information from unauthorized access.

5.2.4.15 Management of Audit Threshold (FMT_MTD.1 (3))

The TSF shall restrict the ability to **modify** the **audit threshold** to **authorized administrators**. FMT_MTD.1.1

5.2.4.16 Management of User Attributes (FMT_MTD.1 (4))

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

Application Note from LSPP: 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 are not within the scope of the LSPP.

5.2.4.17 Management of Authentication Data (FMT_MTD.1 (5))

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: FMT_MTD.1.1

a) authorized administrators; and

b) users authorized to modify their own authentication data.

Application Note from LSPP: 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 users 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.

5.2.4.18 Management of Privileges (FMT_MTD.1 (6))

The TSF shall restrict the ability to **modify** the **privileges for applications** to **authorized administrators**. FMT_MTD.1.1

5.2.4.19 Management of VIOS Mappings (FMT_MTD.1 (7))

The TSF shall restrict the ability to **create, modify, and delete** the**:** FMT_MTD.1.1

- **For Volumes: mappings of logical volumes and physical volumes to VIOS SCSI device drivers acting on behalf of LPAR partitions**
- **For Network: mapping of VIOS Ethernet adapter device drivers to VIOS Ethernet device drivers acting on behalf of groups of LPAR partitions sharing virtual networks**

to **authorized administrators**.

Application Note for AIX: This requirement applies to VIOS only. This requirement is taken directly from CC part 2, not from the LSPP requirements; thus, the LSPP audit requirements do not apply.

5.2.4.20 Revocation of User Attributes (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.1.1

The TSF shall enforce the rules: FMT_REV.1.2

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

d) **Revocations/modifications made by an administrator to security attributes of a user , such as the user identifier, user name, user group(s), user password or user login shell, shall be effective the next time the user logs in. Authorization changes take effect immediately upon reloading the kernel authorization table.**

Application Note from LSPP: 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.

Application Note for AIX: The immediate revocation method that can be used in AIX 5.3 is the one described in the application note from the PP: Make the modifications to the users profile and then force the user to log off.

5.2.4.21 Revocation of Object Attributes (FMT_REV.1 (2))

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 or MAC policies. FMT_REV.1.1

The TSF shall enforce the rules: FMT_REV.1.2

a) The access rights associated with an object shall be enforced when an access check is made;

b) The rules of the Mandatory Access Control policy are enforced on all future operations; and

c) 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 from LSPP: 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.

Application Note for AIX: Immediate revocation for file system objects is not implemented in AIX. AIX uses delayed revocation as described in the application note from the PP.

5.2.4.22 Revocation of VIOS User Attributes (FMT_REV.1(3))

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

The TSF shall enforce the rules: FMT_REV.1.2

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

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

5.2.4.23 Specification of Management Functions (FMT_SMF.1)

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

- **Object security attributes management**
- **User attribute management**
- **Authentication data management**
- **Audit trail management**
- **Audit event management**
- **VIOS mapping management**

Application Note for AIX: This security functional requirement has been added as a result of AIS 32, Final Interpretation 065. This security functional requirement is not included in the LSPP, because the LSPP was developed before AIS 32, Final Interpretation 065 was published. 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.

5.2.4.24 Security Management Roles (FMT_SMR.1(1))

The TSF shall maintain the roles: FMT_SMR.1.1

a) authorized administrators;

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

c) users authorized by the Mandatory Access Control Policy to modify object security attributes;

d) users authorized to modify their own authentication data; and

e) users authorized by the Mandatory Integrity Control Policy to modify object security attributes;

f) users authorized by the TCB Policy to modify object TCB attributes.

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

Application Note for AIX: This requirement applies to AIX, not VIOS.

Application Note for AIX: Authorized administrators are those users of the TOE who have been assigned authorizations to perform individual tasks. The concept of authorizations is discussed in the TSS.

Application Note from LSPP: A LSPP-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 LSPP 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.

5.2.4.25 Security Roles (FMT_SMR.1(2))

The TSF shall maintain the roles**:** FMT_SMR.1.1

a) Prime Administrator

b) System Administrator

c) Development Engineer

d) Service Representative.

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

Application Note for AIX: This requirement applies to VIOS only. This requirement is taken directly from CC part 2, not from the LSPP requirements; thus, the LSPP audit requirements do not apply.

5.2.5 Protection of the TOE Security Functions (FPT)

5.2.5.1 Abstract Machine Testing (FPT_AMT.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. FPT_AMT.1.1

Application Note from LSPP: 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.

Application Note for AIX: Such a test suite is provided as a separate program that an administrator may execute under controlled conditions.

5.2.5.2 Reference Mediation (FPT_RVM.1)

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

Application Note from LSPP: 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.

Application Note for AIX: AIX with PitBull implements a reference monitor as single point of decision within its kernel.

5.2.5.3 Stack Execution Reference Mediation (FPT_RVM.2-AIX)

The TSF shall provide the ability to allow/deny the execution of code residing on the stack of a process created from the executable to anyone who can write to the executable and allow an authorized administrator to override this setting. FPT_RVM.2-AIX.1

5.2.5.4 Domain Separation (FPT_SEP.1)

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

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

Application Note from LSPP: 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.

5.2.5.5 Reliable Time Stamps (FPT_STM.1)

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

Application Note from LSPP: 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.

Application Note for AIX: The reliability of the time stamp is provided by the monotonic increasing time within AIX and the fact that all changes to the time are audited. This allows the determination of the exact sequence of audit events (which according to the application note within the PP is the source for this requirement). The accuracy of the internal clock is on the level of nanoseconds, which allows a precise sorting of audit records according to the time they have been generated.

5.2.5.6 Inter-TSF basic TSF Data Consistency (FPT_TDC.1)

The TSF shall provide the capability to consistently interpret **sensitivity labels** when shared between the TSF and another trusted IT product. FPT_TDC.1.1

The TSF shall use **label encoding rules** when interpreting the TSF data from another trusted IT product. FPT_TDC.1.2

5.2.5.7 TSF Testing (FPT_TST.1)

The TSF shall run a suite of self tests **during initial start-up, and at the request of the authorized administrator** to demonstrate the correct operation of **parts of the TSF**. FPT_TST.1.1(1)

The TSF shall provide authorized users with the capability to verify the integrity of **TSF data**. FPT_TST.1.2 (1)

The TSF shall provide authorized users with the capability to verify the integrity of stored TSF executable code. FPT_TST.1.3 (1)

Application Note for AIX: During boot-time and upon request of authorized administrators, integrity checks is performed to verify that the security attributes of each TCB file have not been modified.

5.3 Strength of Function

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

The security function within the TOE that uses a statistical or probabilistic mechanism is the authentication function that uses passwords.

5.4 TOE Security Assurance Requirements

The target evaluation assurance level for the product is EAL4 [CC] augmented by ALC_FLR.1.

5.5 Security Requirements for the IT Environment

The only IT environment where requirements are stated is the underlying processor, which 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:

The following is a table identifying the SFRs for the IT environment used in this ST, their origin, and the operations performed.

Table 6: SFRs for the IT environment

SFR	Origin
FDP_ACC.1	CC Part 2
FDP_ACF.1	CC Part 2
FMT_MSA.3	CC Part 2
FDP_ACC.1(LPAR)	CC Part 2
FDP_ACF.1(LPAR)	CC Part 2
FIA_UID.2	CC Part 2

Note: Section 4.2 mentions that OE.PROTECT can be implemented with cryptographic controls as one possible security function to meet this objective. But it is 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.

5.5.1 FDP_ACC.1 Subset access control

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

5.5.2 FDP_ACF.1 Security attribute based access control

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

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**. FDP_ACF.1.2

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

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

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.

5.5.3 FMT_MSA.3 Static attribute initialization

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.3.1

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

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 the sc instruction is executed (which effectively causes an interrupt to occur). 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.

5.5.4 FDP_ACC.1 (LPAR) Subset access control

The TSF shall enforce the **LPAR resource access control policy** on **processors, memory regions and I/O slots as objects and partitions as subjects and all access to those resources by a partition**. FDP_ACC.1.1

5.5.5 FDP_ACF.1 (LPAR) Security attribute based access control

The TSF shall enforce the **LPAR resource access control policy** to objects based on **the partition number**. FDP_ACF.1.1

The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed: **a partition shall have access to a processor, a memory region or an I/O slot only if the resource is allocated to the partition by the table in the NVRAM**. FDP_ACF.1.2

The TSF shall explicitly authorize access of subjects to objects based on the following additional rules: **none**. FDP_ACF.1.3

The TSF shall explicitly deny access of subjects to objects based on **no other rules**. FDP_ACF.1.4

5.5.6 FIA_UID.2 User identification before any action

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

Application Note: This identification requirement applies to LPAR partitions being identified by the hypervisor.

5.6 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 system administrators.

6 TOE Summary Specification

6.1 Security Enforcing Components Overview

6.1.1 Introduction

AIX provides a multi-user, multitasking environment, where users interact with the operating system through commands issued to a command interpreter. The command interpreter invokes command programs, which in turn function by making system calls to the operating system kernel. The TSF is comprised of the kernel and trusted processes (trusted programs that are not part of the kernel). All operations performed by users are mediated by the TSF in accordance with the policies defined in Chapter 5.

Within AIX a user can LOGIN to the console of any host computer, request local services at that computer, as well as request network services from any other host in the system.

Processes perform all activity. A process may be started by a user issuing a command, may be created automatically to service a network request, or may be part of the running system created at system initialization. Each process is running a program. A process may begin running a new program (via the exec system call), or create a copy of itself (via the fork system call).

Some activities, such as responding to network requests, are performed directly by the kernel.

The following sections discuss services provided by the kernel, by non-kernel trusted software and the network services. Network services are discussed separately because their implementation is split between kernel and non-kernel components.

The description also contains some supporting functions or mechanism for the security functions. Those are later marked with E_ followed by the name of the function or mechanism. The description of those functions and mechanism then describes, if and how they contribute to satisfy the security functional requirements.

As long as those functions just provide a user interface (e. g. System Management tools) they are not considered to be part of the TSF. But if they directly implement part of a security function (e. g. the trusted processes that reads identification and authentication data) they are considered to be part of the TSF.

6.1.2 Kernel Services

The AIX kernel includes the base kernel and kernel extensions. The base kernel includes support for system initialization, memory management, file and I/O management, process control, audit services and Inter-Process Communications (IPC) services. Kernel extensions and device drivers are separate kernel software modules that perform specific functions within the operating system.

Device drivers are implemented as kernel extensions.

The base kernel has the following key characteristics:

- Can be paged out: Portions of the kernel code and data can be paged out, permitting the kernel to run using less memory than would be required for the whole kernel.
- Pinned: Part of the kernel is always resident or “pinned” into memory and cannot be paged. Pinned code cannot call kernel services that may result in a page fault.
- Can be preempted: The AIX kernel can be preempted. Higher priority threads may interrupt kernel threads, providing support for time critical functions.
- Dynamic and extendible: In standard AIX, kernel extensions can be loaded and unloaded while the system is running to allow a dynamic, extendible kernel without requiring a rebuild and reboot. In the evaluated configuration, dynamic changes to the kernel are prohibited through warnings described in the Security Guide. At system start up, only the kernel extensions that are part of the evaluated product may be loaded. As an example, the administrator can add pieces of hardware (as long as they are part of the hardware configuration listed in this Security Target) to a specific configuration and reboot the system. This will cause the kernel extensions that support the needed device drivers for the new hardware to be loaded. The ability to load/unload kernel extensions is restricted to the processes having the PV_SR_KERNEL privilege.

The AIX 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 VMM manages the paging space used by the AIX file system and provides memory buffers for use within the kernel. The file system and VMM are tightly coupled. Disk pages, whether for file I/O or paging space, are faulted into free pages in memory. The VMM does not maintain a separate pool of pages solely for file system I/O.

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
- Audit event generation

The file and I/O software provides access to files and devices. The AIX Logical File System (LFS) provides a consistent view of multiple physical file system implementations. There are five different types of file systems included in the evaluated configuration: Journaled File System 2 (JFS2), CDROM File System (CDRFS), Universal Disk Format file system (UDFS), Network File System (NFS), and the Special File File System (SPECFS). JFS2, CDRFS and UDFS work off of a physical medium (disk, CDROM, DVD) and NFS works across the network. SPECFS is a file system used internally by the kernel to support disk and other physical and virtual device I/O. The process file system, PROCFS, 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, MAC, MIC, and TCB attributes inferred from the underlying process's security attributes.

cdrfs, udfs, procfs and (client-side) nfs are single level file systems: For mandatory access control, the labels of their mount point apply to all objects in the mounted file system. Single level file systems are not subject to mandatory integrity control, TCB and file security flag policies, and their objects cannot be associated with privileges. This is to be taken into account when reading the following sections of the TSS.

6.1.3 Non-Kernel TSF Services

The non-kernel TSF services are:

- Identification and Authentication services
- Auditing journaling and post-processing services
- Network application layer services
- System integrity checking

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 system administrator and where the kernel prohibits the use/misuse of those tools or commands since they use kernel functions restricted to the system administrator and attempted use by normal users is prohibited by the kernel.

6.1.4 Network Services

Each computer 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 telnet.
- 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 using a reliable byte stream or unreliable datagrams, TCP and UDP respectively.

The higher-level network services are built on TCP or UDP. The TOE supports the TCP application protocols listed below:

- Internet remote login and file transfer services (telnet and ftp) are supported within the evaluated product, as are similar BSD interfaces, including remote command execution (rlogin, rcp, rsh, rexec).
- The Hyper-Text Transfer Protocol (HTTP) is used by the WebInfo document display system (docsearch) for the presentation of public data. The HTTP server is not security relevant and therefore not part of the TSF.
- The Network File System (NFS) protocol is supported for remote file access. This includes some subsidiary protocols, such as the Remote Procedure Call (RPC), portmap protocols, and the mountd protocol for file system import and export.

In addition to the base connectivity provided, all network connections can be configured to support labeling as specified in the CIPSO/RIPSO protocols.

6.1.5 Security Policy Overview

The policy is described as follows:

- A single kernel is running on the system.
- Memory management, segmentation and paging are all handled by the kernel.
- The systems are maintained using a consistent user management policy across all systems.
- Identification and authentication (I&A) is performed locally by the system. 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 (e.g., remote login, and telnet sessions). AIX with PitBull allows authorized administrators to specify executables to be run during login to implement site-specific controls *in addition* to the standard authentication mechanisms provided by the TOE.
- The PID, and its associated TIDs, are unique within the system.
- The names of objects may not be unique within the system; rather, object names are unique on each host. For example, each host maintains its own local file system, but may mount NFS exported file systems at various locations in the local directory tree.
- Discretionary access control (DAC) is based on user identity and group membership. 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 an extended permission list also known as an AIXC Access Control List (ACL) or, in lieu of enforced permission bits, an NFSv4 ACL. Both the extended permissions mechanism and NFSv4 ACL are significant enhancements 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.

- The system supports mandatory access control (MAC) based on sensitivity labels. From a defined set of hierarchical sensitivity levels (SLs), each named object is assigned a dedicated SL, and each user is assigned a range of SLs that he is allowed to access. Users (or, processes acting on behalf of users) can create new objects only with the SL they are currently operating under, cannot read objects that have a higher SL than the user and not write objects that have another SL than themselves. Directories and devices can optionally be assigned a label range – in this case, users can write to an object if their SL is within the SL range of the object.
- The system supports mandatory integrity control (MIC) based on hierarchical integrity labels (TLs). Every named object is assigned a dedicated TL, and each user is assigned a range of valid TLs. Users (or, processes acting on behalf of users) can create objects with the TL that they are currently operating under and cannot write objects that have a higher TL than themselves.
- The system protects trusted computing base (TCB) objects from modification during normal multi-user operation. Objects tagged with the FSF_TCB flag can only be modified when the system is in configuration mode and the user (or, process acting on behalf of a user) has the appropriate privileges to apply changes to the TCB.
- The system uses authorizations and privileges to implement administrative roles and to allow the controlled by-passing of the security policies enforced by the TOE.
- The audit facility generates audit records for activities performed directly by untrusted processes (e.g., the system calls that perform file I/O) as well as trusted process activities (e.g., requests for batch jobs).
- VIOS discretionary access control is performed by VIOS to provide access control between VIOS SCSI device drivers acting on behalf of LPAR partitions and logical/physical volumes. It also provides access control between VIOS Ethernet device drivers acting on behalf of groups of LPAR partitions sharing virtual networks and VIOS Ethernet adapter device drivers.

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 AIX consists of three major components: kernel software, kernel extension 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 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. AIX 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 AIX administrative programs, shells, and standard UNIX utilities that run with administrative privilege, as a consequence of being invoked by a user with proper authorization and with the appropriate MAC privileges. Non-kernel TSF software also includes daemons that provide system services, such as networking and managing audit data, as well as setuid, setgid and privileged programs that can be executed by untrusted users.

6.1.7 TSF Interfaces

Each sub-section here summarizes a class of interfaces in the AIX 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 administrator'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, and 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 present part of the TSF interface.
- Distributed Services, (e.g., telnet) The distributed 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. At a lower level, it allows a system to request a specific service from another system on behalf of a user. Examples of requested services include, executing a command line (e.g.,rsh) or transferring whole files (e.g.,FTP). At the lowest level, it allows a subject on one host in the system to request a connection (e.g.,TCP), or deliver data (e.g.,UDP) to a listening subject. Distributed 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.

6.1.7.2 Operation and Administrator Interface

The primary administrative interfaces to AIX are the same as the interfaces for ordinary users; authorized administrators log into the system using their user ID and password, and perform administrative tasks they have been authorized to perform.

The system is based on a System p5 computer system. The computer may be in one of the following states: shut down, initialization, single-user mode, or multi-user secure state. Administration entails the configuration of the computer as well as the administration of users, groups, files, printers, and other resources within the system.

AIX provides a general purposes, menu-based utility for system administration: *smitty*. Other programs (e.g., */usr/bin/acledit*, */usr/bin/chuser*, */usr/bin/rm*) and scripts are used for system administration, but *smitty* is significant because it provides comprehensive system administration capabilities.

smitty is required for the administration of the AIX system, but the decision as to which administrative utility to use depends upon whether or not the system is in a secure state:

- *smitty* (a cursor-based ASCII version of the System Management Interface Tool (SMIT)) is a text menu interface and dispatcher for a collection of administrative programs.
  smitty is used to administer the local host, i.e., the computer where it is run.

There are other tools for system administration (e. g., msmit) that provide a graphical user interface for system administration. Those tools are not part of the evaluated configuration.

The part of the administrative database 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.

6.1.8 Secure and Non-Secure States

The secure state for the AIX system is defined as a host's entry into multi-user mode with auditing fully operational. At this point, the host accepts user logins and services network requests. 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. The non secure state is also a specific configuration state of the system where kernel security flags and the trusted library path can be modified and FSF_TCB or FSF_TCPROC tagged objects can be created, modified or deleted. This functionality is not available in the operational / multiuser mode

With respect to auditing this Security Target does not define a minimum level of events that need to be audited. But it is required that the system administrator is able to configure all the events mentioned in this Security Target to be included in the audit trail. A system administrator may then define - according to his requirements - define the events that are audited. He is able to change those events using the audit configuration functions during system operation.

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 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
- TOE Protection
- Authorizations
- Mandatory Access Control
- Mandatory Integrity Control
- TCB Access Control implemented with File Security Flags

The TOE Security Functions 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, *telnet* and the *telnetd* 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 AIX system includes all forms of interactive login (e.g., using the Telnet 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 either from a terminal where no user is logged on or when a user that is logged on starts a service that requires additional authentication.

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

Administrators, through the SMIT (*smitty*) administrative interface or via command line tools and editing of configuration files, perform changes to the files that constitute the administrative database.

Users are allowed to change their passwords by using the *passwd* command, which is a *privileged program*. This configuration allows a process running the *passwd* program to read the contents of */etc/security/user* and to modify the

/etc/security/passwd file for the user's password entry, both which would ordinarily be inaccessible to a non-privileged user process. Users are also forced to change their passwords at login time, if the password has expired.

PitBull can be configured to allow user's to create their own passwords or to provide machine-generated passwords. The algorithm in use for generating and choosing a machine-generated password is configurable. User guidelines have been defined for user created passwords that ensure that the probability of guessing a password is less than one in 1,000,000. TOE generated passwords also ensure that the probability of guessing a password is less than one in 1,000,000.

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. The encrypted password of the user itself is not stored in this file but in the file */etc/security/passwd* which is 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/security/passwd* contains the encrypted password, 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.

For a complete list of user attributes see the description of the function SM.4.

The system administrator defines restrictions on authentication data like minimum and maximum size, the minimum number of alphabetic characters, the minimum number of characters that are different from the old password, the minimum number of non-alphabetic characters as well as the maximum life time of a password, the number of unsuccessful login attempts allowed before the account is locked and the times and days the user is allowed to log into the system. Those restrictions can be defined on a per user basis and are stored in the file */etc/security/user*. The system administrator can use those parameters to define a password policy. VIOS supports a subset of these authentication restrictions. See SM.4 for more details.

The file */etc/security/lastlog* contains the time since the last successful login, the time of the last unsuccessful login and the number of unsuccessful login attempts since the last successful login.

6.2.2.2 Common Authentication Mechanism (IA.2)

AIX 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.

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, groups, clearances, default labels, authorizations)

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. A result of success indicated by a 1, or a failure indicated by a 0, is returned to the Terminal State Manager (TSM) program which continues the login process.

When accessing the TOE via its local system console, the ISSO and SO users are exempt from checks pertaining to consecutive authentication failures.

AIX with PitBull allows administrators to augment the authentication mechanism with *additional*, site-specific checks. If the file /etc/IandA is present on a system, the executables listed in this file will be run during login and, if returning a "1", the login processing will fail. However, this mechanism is not supported for login via FTP.

6.2.2.3 Interactive Login and Related Mechanisms (IA.3)

There are eight mechanisms for interactive login and similar activities:

- the standard *login* program (1) for interactive login sessions on the console of a user's local host;
- the telnet protocol (2) protocol for ordinary interactive login sessions on any host in the system;
- the FTP protocol for interactive file transfer (7);
- and the *su* command for changing user identity during a session (8)

All of these mechanisms use the common authentication mechanism described above, but only those that create normal interactive sessions use the standard *login* program; others implement special-purpose types of sessions.

VIOS supports a subset of these mechanisms:

- the standard *login* program
- the *Telnet* protocol
- the *su* command

All those mechanisms will not display a password that is entered via a keyboard for authentication but provide obscured feedback.

6.2.2.3.1 The Login Program

The *login* program establishes interactive user sessions. In AIX and VIOS, *login* is part of the Terminal State Manager (TSM) program. This program prompts for a user identity and authentication (e.g., password), and validates them using the common authentication mechanism described above.

For AIX only, during login, the user can append the option " –e" to his user name, which allows him to specify a label within his clearance to be used during this session (rather than the default label defined for the user).

Authentication prompting may also be suppressed when appropriate (e.g., rsh). If the validation fails, the prompts are repeated until the limits on successive authentication failures are exceeded. Each failure is considered an event that may be audited.

Login establishes a user session as follows:

1. Assigns a session identifier
2. Sets exclusive access for the controlling terminal to the process logging in
3. Calls the common authentication mechanism to check validity of the password provided for the account being accessed, and gains the session security attributes
4. Sets up the user environment
5. Checks for password expiration and if so, prompts for password change
6. The process's user and group identities are changed to those of the user
7. For AIX only, the user's sensitivity clearance is set according to the users entry in the sensitivity clearance database
8. For AIX only, the process's sensitivity label is set to what was specified by the user, provided that the SL is within the user's sensitivity clearance, or to the user's default SL if no SL was specified
9. For AIX only, the process's integrity label is set to the user's default TL
10. For AIX only, the user's integrity clearance is set according to the users entry in the integrity clearance database
11. For AIX only, the user's limiting authorization set, if one has been specified in the LAS database, is applied to the process
12. User is changed to his or her home directory
13. Invokes the user's default shell

The *login* program is always invoked with open file descriptors for the controlling terminal, used when prompting for identity and authentication information, and passes control to the user's shell when the session is created. At this point, the user session is established, the user environment is set up, and the program replaces itself, using the exec system call, with the user's shell).

6.2.2.3.2 Network Login

After an initial login on the console or a terminal, access to other hosts within the same security domain may occur through one of two network protocols: telnet and FTP (refer to section 6.2.2.3.4 for FTP). For AIX, connections are restricted to work only on the same sensitivity clearance. Connection sensitivity clearance labels are not enforced by VIOS.

6.2.2.3.3 Login with telnet

The telnet protocol always requests user identity and authentication by invoking the *login* program, which uses the common authentication mechanism. A user can change identity across a telnet connection if the password for another account is known.

6.2.2.3.4 File transfer using FTP

The FTP protocol is used to create a special type of interactive session that only permits file transfer activities. An FTP session is validated and created directly by the FTP server, which then executes all the user requests directly, as opposed to invoking a user-specified program.

The FTP server invokes the authenticate() function that uses the common authentication mechanism to validate the user identity and password supplied through FTP protocol transactions. User can change identity if password is known.

6.2.2.4 User Identity Changing (IA.4)

Users can change identity (i.e., switch to another identity) using the *su* command. When switching identities, the login UID is not changed, so all actions are ultimately traceable in the audit trail to the originating user. The primary use of the *su* command within AIX is to allow appropriately authorized individuals the ability to assume the root or other administrative identities. In the evaluated configuration the capability to login as the root identity has been eliminated. In the */etc/security/user* file, login to root is set to false for all users and *su* is set to true for administrators. This allows an administrative user to login under his/her real identity, then *su* to the root or other administrative identities.

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

b) When using su to change the id, the authorizations associated with the ID are also changed.

VIOS contains the *su* command, but it doesn't allow users to directly execute it. Instead, the command-line interface will execute a subset of the commands available to a role by using the *su* command under the covers. Commands that would normally allow a user to escape to a shell (i.e., vi) have been modified to disable the shell escape feature. Thus, users cannot directly change their identities during a session.

6.2.2.5 Login Processing (IA.5)

Permissions on the device special files control access to exclusively used public devices. When a user successfully logs in at the local direct attached console, the TSM program changes the ownership of */dev/lft0, /dev/kbd0* and */dev/rcm0* to the login UID of the user and sets the permissions on these devices to be readable and writable by this user. */dev/lft0* is a logical device that provides the users interface to the keyboard and graphics adapter. At system initialization, */dev/lft0* grabs the keyboard and graphics adapter devices. In case of a serially attached ASCII terminal the tty device associated with the terminal changes ownership to the user that is logged in (for example */dev/tty0*)

The */dev/kbd0* device contains two channels for communication between the keyboard and the device driver. Only one channel is active at any given time. The */dev/lft0* device registers for the first channel when the system boots. The second channel is reserved for the X server which is not supported in the TOE. The permissions on the */dev/kbd0* device restrict that only the user who is logged in on the console can access this device. The logged in user could open the second channel, because he/she has permissions. This would redirect the users own keyboard device. This would pose no threat to the operation of the system. The worst thing that would happen is that the login process would not be able to regain access to the */dev/kbd0* device and no other users would be able to login on the console device until the host was rebooted.

The login process executes a revoke to invalidate any open file descriptors for */dev/lft0* or the appropriate */dev/ttyN* device held by a previous user. The revoke call modifies the file descriptors entry in the system open file table, causing further attempts to access the device special file based on that file descriptor to return "bad file descriptor". This ensures that the new login session is isolated from any previous login sessions.

For AIX only, users are assigned a default login SL and TL which is the effective SL and effective TL of the user's process after a successful login. If the user does not want to log in at his/her default login SL, the user may choose to supply a different SL at login time by using the -e option of the login command. The SL supplied by the user must be dominated by the user's clearance and contained in the system accreditation range. The TL cannot be specified by the user at login time. The default login SL and TL are defined in the file /etc/security/clear along with the username and clearance for each user.

For AIX only, the Limiting Authorization Set that controls which authorizations a process can be granted is initialized by the login command as derived from the configuration file in /etc/security/las.

6.2.2.6 Logoff Processing (IA.6)

When a user logs off, all files that were opened by the login shell are closed. Files and devices that were opened by background tasks remain open. However, a background job that had access to the console loses that access prior to the next user's login as stated above.

The ownership of */dev/ttyN*, */dev/lft0*, */dev/kbd0* and */dev/rcm0* is returned to root when the logoff occurs.

6.2.3 Auditing (AU)

This section discusses the implementation of auditing in the evaluated configuration. The data structures and formats are discussed first, followed by how audit is controlled, a description of bin mode auditing, the programs used to post process the audit data, the programs used to review audit data, audit file protection, and finally the potential for audit data loss. AIX includes tools for pre-selecting and post-selecting audit data, viewing audit trails, and merging multiple audit trails into one file.

6.2.3.1 Audit Record Format (AU.1)

The audit record consists of a header that contains information identifying the user and process who generated the record, the status of the event (success or failure), and the CPU ID for the system. The CPU ID field allows the administrator to differentiate between individual machines when merging the contents of multiple audit trails. An optional variable length tail contains extra information about the event, as defined in */etc/security/audit/events*.

The audit record is a fixed length record that contains information about the user who caused the event and whether the event was created due to a success or failure. The audit record is defined in */usr/include/sys/audit.h.*

Table 7: Audit Record Format

Magic number for audit record.	
The length of the tail portion of the audit record.	
The name of the event and a null terminator.	
An indication of whether the event describes a successful	
operation. The values for this field are:	
0	Indicates successful completion.
1	Indicates a failure.
>1	An errno value describing the failure.
The real user ID; that is, the ID number of the user who created the process that wrote this record.	
The login ID of the user who created the process that wrote this record.	
The program name of the process, along with a null terminator.	
The process ID of the process that wrote this record.	
The process ID of the parent of this process.	
The thread ID.	
The time in seconds at which this audit record was written.	
The nanoseconds offset from time. (used during bin recovery and trail merging to ensure proper record ordering)	
CPU identifier.	

For File system objects the following information is added:

Sensitivity Label
Max Sensitivity Label (Dirs, Devs)
Information Label
Integrity Label
Inheritable Privs from process
Privileges to be added to process
Privileges for authorized process
Access Authorizations
Privileged Authorizations
Capability Sets
File System Security Flags

For processes the audit records have the following additional information:

Network Identifier
Security Flags
Sensitivity Label
Min SL Clearance Label
Max SL Clearance Label
Information Label
Integrity Label
Effective Privilege Vector
Maximum Privilege Vector
Limiting Privilege Vector
Used Privilege Vector
Capability Set
Limiting authorization Set
Process Identifier
Min TL Clearance Label
Max TL Clearance Label

6.2.3.2 Audit Record Generation (AU.2)

Audit record generation begins with the detection of an event, and follows the record as it advances to storage.

Event detection is distributed throughout the TSF, both in kernel and user mode. Programs and kernel modules that detect events that may be audited are responsible for reporting these events to the system audit logger. The system audit logger is part of the kernel, and can be accessed via a system call for trusted program auditing, or via a kernel procedure call for supervisor state auditing.

The audit logger is responsible for constructing the complete audit record, including the identity and state information and the event specific information. The audit logger appends the record to the active bin. A bin is a file that is used to store raw audit records before they are processed and stored in the audit trail.

6.2.3.3 Audit Record Processing (AU.3)

Audit record processing includes a description of bin mode auditing and the backend processors that are utilized by the audit subsystem.

6.2.3.3.1 Bin Mode Auditing

When Bin mode auditing starts, two separate bin files are allocated to store raw audit records by the auditbin daemon. When one bin file fills, the daemon switches to the other bin file and invokes the processing command specified in */etc/security/audit/bincmds* to empty the full cache file.

When that operation is complete, auditbin notifies the kernel that it is permitted to reuse the cache file. This mechanism of switching and emptying audit bins continues so long as auditing is enabled. The size a bin file may reach before being considered full is defined in */etc/security/audit/config*.

A bin file begins with a header. The tail is written when the audit bin is switched or when auditing is shut down.

6.2.3.3.2 Backend Audit Processors

There are two backend utilities available for use: *auditcat* and *auditselect*. The backend processor writes the raw audit records to the system audit trail or to a specified file after manipulating them.

Bin mode auditing makes use of *auditcat* and *auditselect*. The result of *auditcat* or *auditselect* can be directed to a file for permanent audit storage.

6.2.3.3.2.1 *auditcat*

The *auditcat* command reads audit records from standard input or from a file, and processes the records and sends them to standard output or to the system audit trail.

6.2.3.3.2.2 *auditselect*

The *auditselect* command can be used as both a pre-processing and post-processing tool. As a pre-processing tool, the *auditselect* command serves the same purpose as *auditcat*, but adds the ability to specify conditions that an audit record must meet. This allows a system to be configured to save audit records that relate to login in one file, and audit records that relate to file access in a separate file.

Auditselect utilizes an expression to apply against the current audit record. The expression consists of one or more terms joined by the logical operators && (and), || (or) and ! (not). Each term in the expression describes a field, a relational operator and a value.

The following is an example expression to select all the *FILE_Open* events:

```
event==FILE_Open
```

The event field identifies that *auditselect* should query based on the name of the event. The operator is equal and the name of the event is *FILE_Open*.

Table 8: Available Fields. The available fields are used to build expressions with *auditselect*.

Field	Definition
event	Name of the audit event
command result	Status of the audit event. The value of the result field must be one of the following: OK, FAIL, FAIL_PRIV, FAIL_AUTH, FAIL_ACCESS, RFM_MAC_Fail, or FAIL_DAC. FAIL matches all other error codes.
login	ID of the login user of the process that generated the audit event.
Real	ID of the real user of the process that generated the audit event.
Pid	ID of the process that generated the audit event.
Ppid	ID of the parent of the process that generated the audit event.
Tid	ID of the kernel thread that generated the event.
Time	Time of day the audit event was generated.
Date	Date the audit event was generated.
Host	Hostname of the machine that generated the record. The reserved name UNKNOWN can be used to match any machines that are not listed in the /etc/security/audit/hosts file.
subj_SL	SL of the subject at the time of the audit event.
obj_SL	SL of the object at the time of the audit event.
mac_pass	If successful MAC event.
mac_fail	If failed MAC event.
mic_fail	If successful MIC event
mic_fail	If failed MIC event

Field	Definition
dac_pass	If successful DAC event
dac_fail	If failed DAC event
priv_pass	If successful PV event
priv_fail	If failed PV event
fsf_pass	If successful FSF event
fsf_fail	If failed FSF event
Auth	Auth event

Auditselect allows to selectively extract individual audit records.

6.2.3.4 Audit Review (AU.4)

Two different commands exist for the review of audit records in the system: *auditpr* and *auditselect*.

The *auditpr* command formats audit records to a display device or to a printer for review. The *auditpr* command also allows the administrator to select which of the fields to include in the output as well as the order to display them. The fields available for inclusion with the output of the *auditpr* command are listed in the following table.

Table 9: Available Fields from *auditpr*. These fields are available for output from *auditpr*.

audit event	user's login name	event status	time the record was written	command name	real user name	process ID	ID of the parent process	kernel thread ID	name of the host that generated the audit record	event specific tail data

The default values are the audit event, the user's login name, the audit status, the kernel thread ID and the command name *auditselect* allows the administrator to build an expression that will be applied to the stored audit records. The details of the *auditselect* command are listed in section 6.2.3.3, Audit Record Processing.

The *auditmerge* command provides a method of combining multiple audit trail files into a single audit trail file. These multiple files can come from different hosts, providing a centralized audit analysis function. As the two files are processed, the record with the oldest time stamp that still remains is written into the audit trail. This process continues until there are no more audit records to process. The Security Guide directs the system administrator to transfer the audit files to be merged to the same host.

The commands *auditpr* and *auditmerge* allow an authorized administrator to read the audit records and convert them to human readable.

6.2.3.5 Audit File Protection (AU.5)

The audit trail files, configuration files, bin files, and the /audit directory are protected on each system using normal file system permissions. Each audit file grants read access to the root user and the audit group, and write access to only the root user. The AUDIT authorization is needed for the audit programs that are used to read these files.

6.2.3.6 Audit Record Loss Prevention (AU.6)

Bin mode auditing is susceptible to the exhaustion of disk space available to the /audit directory or to a system crash. In the case of a system crash, all data in physical memory is lost, including any audit records that had not yet been flushed to disk. The audit subsystem enforces a 32K byte limit on the size of an individual audit record, and only one audit record can be in transit between a thread and the kernel at any given time. When the system is no longer able to write audit records to the audit bins either the system will stop in "panic" mode or a counter will show the number of audit records lost. This counter is written in an audit record the next time the system is able to produce audit records again. If the TOE stops in case it is unable to write audit records or if the TOE just counts the number of audit record lost is a configuration parameter that can be set by the System Administrator.

The AIX 5.3 Security Guide includes instructions to the administrator to back up all files, including audit data, on a regular basis to avoid the loss of data due to hard disk failures.

6.2.3.6.1 Audit Record Loss Prevention for Bin Mode Auditing

As a new feature of AIX 5.3 the system allows an administrator to define a threshold value for the amount of free space in the file system holding the audit files. When the amount of free space in this file system is below this defined threshold value this fact will be reported to an administrator. This allows the administrator to take the appropriate actions to prevent the system to enter the panic mode due to the inability to write events to the audit trail.

AIX 5.3 provides a panic mode for use with bin mode auditing. The panic mode option halts the host when the current audit bin stops accepting additional records, preventing the unnecessary loss of audit records. This only occurs with the exhaustion of disk space. The AIX 5.3 Security Guide contains instructions for enabling panic mode, as panic mode is not enabled by default.

The result of halting the system because panic mode was invoked would be the loss of any audit data presently in the host's memory that had not been written to disk. In addition, audit records could be lost for operations that were underway but had not yet completed generating audit records. This minimizes the damage caused by the lack of disk space, because only the audit records that are currently in memory are lost.

A recovery process for audit bins exists in the evaluated configuration. If either of the bin files is not empty when audit is started, the *auditbin* daemon executes the bin mode post-processing command to process the bins.

The amount of audit data that can be lost in bin mode is minimized by the use of the binsize and bytethreshold parameters in the */etc/security/audit/config* file. The binsize parameter sets the maximum size a bin may reach before the auditbin daemon switches to the other bin, and executes the bin mode post-processing command. The bytethreshold parameter sets the amount of data in bytes that is written to a bin before a synchronous update is performed. The AIX 5.3 Security Guide states that the binsize and bytethreshold parameters should be set to 64K bytes each to minimize audit data loss. The amount of audit data that could be lost due to a failure in bin mode is the combination of these two files, or 128K bytes.

6.2.3.7 Audit System Privileges (AU.7)

The enforcement of the TOE's auditing policies is supported, in addition to the general access control policies (DAC, MAC, MIC), by the following privileges, allowing a process to

1. perform restricted operations (start/stop of accounting mechanism) pertaining to the accounting subsystem (PV_AU_ACCT);
2. record/add audit records (PV_AU_ADD);
3. turn on/off auditing or change audit system configuration (PV_AU_ADMIN);
4. query the status of the audit system or the audit mask of a process (PV_AU_GETINFO);
5. read a file marked as an audit file (PV_AU_READ);
6. write or delete a file marked as an audit file, or mark a file as an audit file (PV_AU_WRITE);
7. obtain all privileges listed in 1. to 6. (PV_AU).

See TP.9 for a description of file security flags that influence the behavior of the audit subsystem.

6.2.4 Discretionary Access Control (DA)

This section outlines the general DAC policy in AIX as implemented for resources. A subset of these resources are file system objects where access is controlled by one of two policies (i.e., a file system object can only have one policy associated with it at a time):

- AIXC policy – the AIX classic access control policy
- NFSv4 policy – the Network File System version 4 (NFSv4) access control policy

The AIXC policy uses permission bits and, optionally, extended permissions. The extended permissions are in the form of an access control list (ACL) where each entry in the ACL can define the permissions of a specific user or group. This is described in more detail in the following sections.

The NFSv4 policy uses fine grained permissions. The fine grained permissions are in the form of an ACL where each entry in the ACL can enable a number of fine grained permissions for a user, group, or for everyone. This is described in more detail in the following sections.

Permission bits are the standard UNIX DAC mechanism and are used on all AIX file system named objects. 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 and fine grained permission mechanisms are supported only for file system objects and provide a finer level of granularity than do permission bits.

The policies for all resources are 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".

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.

The privilege PV_DAC will also grant full access regardless of the setting of permission bits or ACLs. A subset of PV_DAC can be used for explicit overrides, for example PV_DAC_R_NS to override DAC read restrictions on non system files or PV_DAC_W to override DAC restrictions on any file.

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

AIXC Permission-bit and Extended Permission Policy

A subject whose effective UID matches the file owner ID can change the file attributes, the base permissions, and the extended permissions. Changes to the file group are restricted to the owner.

For new files, the group identifier must either be the current effective group identifier or one of the group identifiers in the concurrent group set.

NFSv4 Policy

A subject whose effective UID matches the file owner ID can change the file attributes, the base permissions, and the fine grained permissions. (If an object has an NFSv4 ACL, the base permissions (excluding the setuid, setgid,and save text bits) are ignored when making access decisions, but they are set to approximate the value of the ACL.) Additional rules regarding who can manage NFSv4 ACLs and object attributes are provided in section 6.2.4.3.1.2.2.

For new files, the group identifier must either be the current effective group identifier or one of the group identifiers in the concurrent group set.

The NFSv4 policy allows for file system objects to inherit ACL entries from the parent directory's ACL. Subdirectories can inherit different entries than other file system objects. The ability to propagate the ACL entries to subdirectories can be limited to just the subdirectories within the parent directory.

6.2.4.1 Permission Bits (DA.1)

AIX uses standard UNIX permission bits to provide one form of DAC for file system named objects. 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). 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 When a process attempts to reference an object protected only by permission bits, the access is determined as follows:

- Effective UID = Object's owning UID and the owning user permission bits allow the type of access requested. Access is granted with no further checks.

- Effective GID, or any supplementary groups of the process = Object's owning GID, and the owning group permission bits allow the type of access requested. Access is granted 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 does not possess the needed PV_DAC privilege or the appropriate subset of PV_DAC then the access attempt is denied.

As a special case that has been modeled as part of the DAC Policy in this Security Target, a read-only bit for file systems can be set upon mount time, yielding the denial of every write request for the file system.

6.2.4.2 Extended Permissions (DA.2)

6.2.4.2.1 AIXC Extended Permissions

The extended permissions consist of an essentially unlimited number of additional permissions and restrictions for specific users and groups. Each entry in the extended permissions list consists of three parts: an entry type, a set of permissions, and an identity list.

- The entry type is the value permit, deny, or specify (indicating that the entry indicates a set of permissions to be allowed as supplemental to the listed identity(ies), denied to the listed identity(ies), or that the permissions permitted and the complementary set denied to the listed identity(ies) respectively).
- The permission set is zero or more of the permissions read, write, and execute.
- The identity list is one or more values specifying users and/or groups. The entry is applied if the process' effective UID, effective GID, and supplemental groups match all values in the list. The term "match" means that for each value in the identity list, if the value is for a UID, that the specified UID is the same as the process' effective UID, and if the value is for a GID, that the specified GID is either the same as the process' effective GID or the specified GID is included in the process' list of supplemental GIDs.

There is no explicit ordering of entries within the extended permissions. To determine access rights, the kernel takes into account all entries that match the UID or GID of the process. For each entry, the permit and specify bits are added to a permissions list and the deny and bitwise negation of the specify are added to a restrictions list. The restrictions are bitwise removed from the permissions and the resulting list is used in the access determination.

The maximum size for the extended permissions is one memory page (4096 bytes). The entries are variable length. Each entry takes a minimum of 12 bytes (two for the length of the entry, two for the permission type and permissions allowed, two for the number of identity entries, two for the type of identity entry, and four for each UID/GID). As a result, there can be over 300 entries in an extended permissions list, which is in practice unlimited.

Collectively, the file attributes, base permissions, extended permissions, and extended attributes are known as the file AIXC Access Control List (ACL). AIXC ACLs have a textual representation (used with commands such as *aclget*) and binary representations (for storage in the file system).

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:

> 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 type of access is within the union of all permission rights (grant entries) defined in the access control list of the object for the subject and is not within the logical union of all restrictions (deny entries) defined in the access control list of the object for the subject. If no entry in the extended permissions either allows or denies access, the access right defined in the permission bits apply. In any other case access is denied.

6.2.4.2.2 NFSv4 Fine Grained Permissions

An NFSv4 ACL consists of a list of entries with the following fields:

- Type Field – This field contains one of the following values:
  - ALLOW – Grants the subject, specified in the Who field, the permission(s) specified in the Mask field.
  - DENY – Denies the subject, specified in the Who field, the permission(s) specified in the Mask field.

- Mask Field – This field contains one or more of the following fine grained permission values:
    - READ_DATA / LIST_DIRECTORY – Read the data from a non-directory object or list the objects in a directory.
    - WRITE_DATA / ADD_FILE – Write data into a non-directory object or add a non-directory object to a directory.
    - APPEND_DATA / ADD_SUBDIRECTORY – Append data into a non-directory object or add a subdirectory to a directory.
    - READ_NAMED_ATTRS – Read the named attributes of an object. (There are no named attributes.)
    - WRITE_NAMED_ATTRS – Write the named attributes of an object. (There are no named attributes.)
    - EXECUTE – Execute a file or traverse/search a directory.
    - DELETE_CHILD – Delete a file or directory within a directory. (Applies to directories.)
    - READ_ATTRIBUTES – Read the basic (non-ACL) attributes of a file.
    - WRITE_ATTRIBUTES – Change the times associated with a file or directory.
    - DELETE – Delete a file or directory.
    - READ_ACL – Read the ACL.
    - WRITE_ACL – Write the ACL.
    - WRITE_OWNER – Change the owner and group.
    - SYNCHRONIZE – Synchronize access. (Exists for compatibility with other NFSv4 clients, but has no implemented function.)
- Flags Field – This field defines the inheritance capabilities of directory ACLs and indicates whether the Who field contains a group or not. The field contains zero or more of the following flags:
    - FILE_INHERIT – Specifies that, in this directory, newly created non-directory objects will inherit this entry.
    - DIRECTORY_INHERIT – Specifies that, in this directory, newly created subdirectories will inherit this entry.
    - NO_PROPAGATE_INHERIT – Specifies that, in this directory, newly created subdirectories will inherit this entry, but these subdirectories will not pass this entry to their newly created subdirectories.
    - INHERIT_ONLY – Specifies that this entry does not apply to this directory, only to the newly created objects that inherit this entry.
    - IDENTIFIER_GROUP – Specifies that the Who field represents a group; otherwise, the Who field represents a user or a special Who value.
- Who Field – This field contains one of the following values:
    - User – Specifies the user that this entry applies to.
    - Group – Specifies the group that this entry applies to.
    - Special – This attribute can be one of the following values:
        - OWNER@ - Specifies that this entry applies to the owner of the object
        - GROUP@ - Specifies that this entry applies to the owning group of the object.
        - EVERYONE@ - Specifies that this entry applies to all users of the system *including* the owner and group.

If the ACL is empty, the PV_DAC_R privilege is required to access the object.

The owner of an object implicitly has the following mask values regardless of what the ACL may or may not contain:

- READ_ACL

- WRITE_ACL
- READ_ATTRIBUTES
- WRITE_ATTRIBUTES

APPEND_DATA is implemented as WRITE_DATA. Effectively, there's no functional distinction between WRITE_DATA and APPEND_DATA. Both values must be set or unset in unison which is enforced by the TOE.

Object ownership can be modified through the use of WRITE_OWNER. Section 6.2.4.3.1.2.2 details how WRITE_OWNER works. When the owner is changed, the setuid bit is turned off. When the group is changed, the setgid bit is turned off.

The inheritance flags only have meaning in a directory's ACL and only apply to objects that are created in the directory after the inheritance flags have been set (i.e., existing objects are not affected by inheritance changes to the parent directory's ACL).

The entries in an NFSv4 ACL are order dependent. To determine if the requested access is allowed, each entry is processed in order. Only entries which have a Who field that matches the effective UID, if a user is specified in the entry, or effective GID, if a group is specified in the entry, of the subject are considered. Each entry is processed until all of the bits of the requester's access have been ALLOWED. Once an access type has been ALLOWED by an entry, it is no longer considered in the processing of later entries. If a DENY entry is encountered where the requester's access for that mask value is necessary and undetermined, the request is denied. If the evaluation reaches the end of the ACL, the request is denied.

The maximum supported ACL size is 64KB. Each entry in an ACL is of variable length and 64KB is the only limit on an entry.

6.2.4.3 Discretionary Access Control: File System Objects (DA.3)

The Discretionary Access Control (DAC) policy is described above. This section describes the details of DAC policies as they apply to file system objects.

6.2.4.3.1 Common File System Access Control

This section describes the common DAC policy applied to file system objects, including policies for object contents and attributes.

6.2.4.3.1.1 *DAC Contents Policy*

6.2.4.3.1.1.1 AIXC Permission-bit and Extended Permissions Contents Policy

The permission-bit and ACL DAC policy determines the effective access that a process may have to the contents of a file system object: some combination of read(r), write (w), and execute (x). In general, read access permits the object's contents to be read by a process, and write permits them to be written; execute is interpreted differently for different object types. Some object types (unnamed pipes, symbolic links) do not use the permission bits at all.

6.2.4.3.1.1.2 NFSv4 Contents Policy

The NFSv4 policy determines the effective access that a process may have to the contents of a file system object. How this policy works is described in DA.2. Some object types (unnamed pipes, symbolic links) do not use the NFSv4 policy at all. The permission bits (excluding the setuid, setgid, and save text bits), specifically the user/group/other bits, are ignored when making access control decisions if an NFSv4 ACL exists on the object.

6.2.4.3.1.2 *DAC Attributes Policy*

6.2.4.3.1.2.1 AIXC Permission-bit and Extended Permissions Contents Policy

In general, a process must be the object's owner, or have privilege, to change the objects attributes, and there are no DAC restrictions on viewing the attributes, so any process may view them. However, the following are exceptions to the rule:

- The permission bits and ACL (permission bits, extended permissions and attributes) of an object may be changed by an owner or by a subject having the PV_DAC privilege or the appropriate subset of PV_DAC.

- The owning group ID of an object may be changed by an owner, but only to a group of which the process is currently a member, unless it has PV_DAC or the appropriate subset of PV_DAC.
- The owning user ID of an object may only be changed by an administrator with PV_DAC or the appropriate subset of PV_DAC.

6.2.4.3.1.2.2 NFSv4 Contents Policy

The NFSv4 policy provides control over who can read and write the attributes of an object. A subject with the PV_DAC privilege can always override the NFSv4 policy. The object owner can allow others to read and write the attributes of an object using the READ_ATTRIBUTES, WRITE_ATTRIBUTES, READ_NAMED_ATTRS, and WRITE_NAME_ATTRS attributes of the ACL mask. The owner can control who can read and write the ACL using the READ_ACL and WRITE_ACL attributes of the ACL mask. The object owner always has READ_ATTRIBUTES, WRITE_ATTRIBUTES, READ_ACL, and WRITE_ACL access. The owner can also allow others to change the owner and group of the object using the WRITE_OWNER attribute. An object owner cannot change the owner or group of the object by default, but the owner can add a WRITE_OWNER entry to the ACL specifying themselves, or the object could inherit an ACL entry which specifies a WRITE_OWNER entry with a Who value of OWNER@.

6.2.4.3.1.3 *DAC Defaults*

6.2.4.3.1.3.1 AIXC Permission-bit and Extended Permissions Defaults

The default access control on newly created FSOs is determined by the permissions associated with the directory where the FSO was created, the effective user ID, group ID, and umask value of the process that created the FSO, and the specific permissions requested by the program creating the FSO.

- The owning user of a newly created FSO will be the effective UID of the creating process.
- If the setgid bit is set on the containing directory, then the owning group of a newly created FSO will be the owning group of the containing directory. If the setgid bit is not set on the containing directory, then the owning group of the newly created FSO will be the effective GID of the creating process.
- The initial access permissions on the FSO are those specified by the creating process bit- wise ANDed with the one's complement of the umask value. For example, if a program specified initial permissions of 0664 (read/write for owner, read/write for group, and read for world) but the umask value were set to 0027 (prevent write for group or world, prevent all permissions for world), then the initial file permissions would be set to 0640 (or 0644 bit-and 0750).
- There are initially no extended permissions associated with an FSO. Extended permissions can be set by applications or by users using AIX commands.

Base and extended access permissions can be changed by any process with an effective UID equal to the owning UID of the FSO, providing that the effective UID has at least the execute permission to the containing directory. Note that since a file may have multiple hard links, the process can use any of the containing directories (e.g., if there is any directory containing a link to the file, then that path could be used as a means to get to the file and change its permissions).

6.2.4.3.1.3.2 NFSv4 Defaults

If the parent directory does not have any NFSv4 inheritance entries applicable to the FSO being created, then the FSO will be created using the AIXC defaults mentioned above. Otherwise, the parent directory's inheritance entries will be copied into and become the ACL of the newly created FSO as per the rules of NFSv4 inheritance. NFSv4 inheritance is described in DA.2.

- The owning user of a newly created FSO will be the effective UID of the creating process.
- If the setgid bit is set on the containing directory, then the owning group of a newly created FSO will be the owning group of the containing directory. If the setgid bit is not set on the containing directory, then the owning group of the newly created FSO will be the effective GID of the creating process.

The permission bits are set on the object to approximate the values contained in the ACL.

6.2.4.3.1.4 *DAC Revocation on File System Objects*

With the exception of NFS objects, file system objects (FSOs) access checks are performed when the FSO is initially opened, and are not checked on each subsequent access. Changes to access controls (e.g., revocation) are effective with the next attempt to open the FSO.

For NFS objects, access is checked for each operation. A change to the access rights for an NFS FSO take effect as of the next NFS request.

In cases where the administrator determines that immediate revocation of access to an FSO is required, the administrator can reboot the computer, resulting in a close on the FSO and forcing an open of the FSO on system reboot. This method is described in the AIX 5.3 Security Guide.

6.2.4.3.2 DAC: Ordinary File

Ordinary files support the concept of execution. Execute access is required to execute the file as a program or script. When an executable file has the set-user-ID or set-group-ID flags set, and the file owner or file group is not the same as the process's current effective user-ID or group-ID the executing program changes the process's security attributes. Otherwise the attributes remain unchanged. AIX doesn't support set-UID or set-GID scripts.

6.2.4.3.3 DAC: Directory

The execute access for directories governs the ability to traverse the directory as part of a pathname. A process must have execute access in order to traverse the directory during pathname resolution.

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.

6.2.4.3.4 DAC: UNIX Domain Socket Special File

UNIX domain socket files are treated as files in the AIX 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 both read and write access to the socket file.

UNIX domain sockets exist in the file system name space. The socket files are supported by both the AIXC and NFSv4 policies.

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

6.2.4.3.5 DAC: Named Pipes

Named pipes are treated identically to any other file in the AIX file system from the perspective of access control. Therefore both AIXC and NFSv4 policies are supported by named pipes. 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).

6.2.4.3.6 DAC: Device Special File

The access control scheme described for FSOs is used for protection of character and block device special files. The DAC settings on 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.

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 (e.g., revocation) are effective with the next attempt to attach to the SMS.

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

This method is the described in the Security Guide.

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).

However, once a SMS has been marked as deleted, additional processes cannot attach to the SMS and it cannot be undeleted.

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.

- The owning user and creating user of a newly created SMS will be the effective UID of the creating process.
- The owning group and creating group of a newly created SMS will be the effective GID of the creating process.
- 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.
- SMSs do not have extended permissions.
- SMSs do not support NFSv4 ACLs.

Access permissions can be changed by any process with an effective UID equal to the owning UID or creating UID of the SMS.

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). Changes to access controls (e.g., revocation) are effective upon the next request for access. 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.

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

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 UID of the creating process.
- The owning group and creating group of a newly created message queue will be the effective GID 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 have extended permissions.
- Message queues do not support NFSv4 ACLs.

Access permissions can be changed by any process with an effective UID equal to the owning UID or creating UID of the message queue. Access permissions can also be changed by any process having the appropriate privilege.

6.2.4.4.3 DAC: Semaphores

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

In cases where the administrator determines that immediate revocation of access to a SysV semaphore is required, the administrator can reboot the computer, thus destroying the semaphore and any processes waiting for it. This method is described in the Security Guide.

If a process requests deletion of a SysV 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). However, once a SysV semaphore has been marked as deleted, additional processes cannot perform semaphore operations and it cannot be undeleted.

The default access control on newly created SysV 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.

- The owning user and creating user of a newly created SysV semaphore will be the effective UID of the creating process.
- The owning group and creating group of a newly created SysV semaphore will be the effective GID of the creating process.
- The initial access permissions on the SysV semaphore must be specified by the creating process, or they are set to null and the object is inaccessible until the owner sets them.
- SysV semaphores do not have extended permissions.
- SysV semaphores do not support NFSv4 ACLS.

Access permissions can be changed by any process with an effective UID equal to the owning UID or creating UID of the semaphore, and can be overridden by privileges.

No security claims are made for non-SysV semaphores.

6.2.4.5 Discretionary Access Control: VIOS (DA.5)

VIOS resides in a separate LPAR partition and provides basic discretionary access control between VIOS SCSI device drivers acting on behalf of LPAR partitions and SCSI-based logical volumes and physical volumes through mappings. An LPAR partition (via a VIOS SCSI device driver) may be mapped to 0 or more logical and physical volumes, but a volume can only be mapped to at most one LPAR partition. This mapping limits an LPAR partition to only the volumes assigned to it.

VIOS also controls the mapping of VIOS Ethernet adapter device drivers to VIOS Ethernet device drivers acting on behalf of groups of LPAR partitions sharing a virtual network. In the evaluated configuration, only a one-to-one mapping of an Ethernet adapter device driver to an Ethernet device driver acting on behalf of a group of LPAR partitions is allowed. The one-to-one mapping is configured by the administrator and enforced by the device drivers. Also, the Ethernet packets must not be tagged with a VLAN tag in the evaluated configuration. This mechanism can be used to limit which LPAR partitions see certain Ethernet packets.

VIOS is restricted to administrator access only. VIOS allows all administrative roles except the Service Representative roles to manage the access control mechanisms previously mentioned.

6.2.5 Privileges (PV)

A privilege is an attribute of a process that allows the process to bypass specific restrictions and limitations of the system (DAC, MAC, MIC, TCB). Privileges are used to override security constraints, to permit expanded use of certain system resources such as memory and disk space, and to adjust the performance and priority of the process. In addition, privileges can be used directly within a user-level program that is responsible for mediating or enforcing security.

The privileges used on the PitBull Foundation extension for AIX allow the administration of the system without the use of the all powerful root administrator ID by granting explicit privileges to users (via authorizations) or commands.

6.2.5.1 Identification of privileges (PV.1)

Privilege sets can be attached to both subjects and objects, but object privilege sets are used only on executable files and only for modifying the process privilege sets when the file is executed.

The privileges enforced by the TOE's reference monitor are as follows:

- Audit privileges (identified in section 6.2.3.7)
- Authorization privileges (identified in FDP_ACF.1(3))
- DAC privileges (identified in FDP_ACF.1(1))

- File system privileges, allowing a process to
  a. perform the mknod system call (PV_FS_MKNOD);
  b. use the mount and umount system calls (PV_FS_MOUNT);
- Label privileges, allowing a process to
  a. modify subject sensitivity clearance labels, subject to the process's clearance (PV_CL);
  b. modify subject integrity clearance labels, subject to the process's clearance (PV_CL_TL);
  c. read the labeling database (PV_LEF);
  d. downgrade SL, subject to the process's clearance (PV_SL_DG);
  e. upgrade SL, subject to the process's clearance (PV_SL_UG);
  f. downgrade the SL of a packet being placed on a STREAM, subject to the process's clearance (PV_SL_DG_STR);
  g. upgrade the SL of a packet being placed on a STREAM, subject to the process's clearance (PV_SL_UG_STR);
  h. change object SL, subject to the process's clearance (PV_SL_FILE);
  i. change subject SL, subject to the process's clearance (PV_SL_PROC);
  j. change its own CL, subject to the process's clearance (PV_SL_SELF);
  k. modify subject and object TLs (PV_TL);
- MAC privileges (identified in FDP_IFF.2(1))
- MIC privileges (identified in FDP_IFF.2(3))
- Network/Driver/STREAMS privileges (identified in FDP_IFF.2(2))
- Privilege privileges, allowing a process to
  a. change the IPS or PPS of a file (PV_PV_FILE);
  b. change its own MPS or change the MPS or EPS of another process (PV_PV_PROC);
- System resource privileges, allowing a process to
  a. perform restricted file system operations (PV_SR_FS);
  b. perform restricted operations involving low-level hardware operations (PV_SR_HW);
  c. modify kernel configuration and resource limits, load/unload kernel extensions (PV_SR_KERNEL);
  d. change resource limits and restrictions for its own or another process to which it has write access (PV_SR_PROC);
  e. change its root directory (PV_SR_PROC_CHRT);
  f. bypass system resource limits (PV_RAC_PROC_BYPASS);
  g. set system resource limits (PV_RAC_PROC_SET);
  h. alter its environment (PV_PROC_ENV);
  i. configure and write RAS records (PV_RAS_CONFIG);
  j. obtain all privileges listed in a. to e. (PV_SR);
- Super user privileges, allowing a process to
  a. Obtain all authorizations associated with the AZ_ROOT authorization (PV_SU_AZ);
  b. Obtain all privileges associated with a standard super user regardless of the UID (PV_SU_ROOT);
  c. Obtain all privileges associated with a standard super user when its UID is 0 (PV_SU_EMUL);
  d. Cause the getuid(2) system call to return 0 (PV_SU_UID);

e. Obtain all privileges listed in a. to d. (PV_SU).

- TCB privilege (identified in FDP_ACF.1(2))
- X Window (GUI) privileges (PV_X_ROOT) – not relevant for the TOE!
- Miscellaneous privileges, allowing a process to
  a. change to REAL mode (i.e. be able to see all the real contents of a partitioned directory as well as the partitioned sub-directories to which it has access (PV_PDMODE);
  b. obtain the equivalent of all other privileges except PV_SU (and the privileges it dominates) and PV_X_ROOT (PV_ROOT);
  c. change the kernel security configuration flags (PV_SECCONFIG);
  d. perform actions limited to trusted path processes (PV_TP);
  e. set or clear the kernel trusted path flag (PV_TP_SET).

Privileges are hierarchical so that for example the privilege PV_DAC is a superset of all the PV_DAC_* privileges. The mechanisms of inheritance and dominance are explained in [PB_AIXARCH].

6.2.5.2 Process Privilege Sets (PV.2)

There are three types of process privilege sets: *effective privilege* sets (EPS), *maximum privilege* sets (MPS), and *limiting privilege* sets (LPS).

The EPS is used to actually override system restrictions. A process can add or remove privileges from its own EPS subject to the limitations imposed by the MPS.

The MPS is the set of privileges over which a process has control. The MPS is always a superset of the process's EPS. A process can always remove a privilege from its MPS. A process's MPS can only be increased if the process has the appropriate privilege, and even then it is restricted by the LPS of the process. The MPS of a process can also be modified when the process runs an executable file, but this too is limited by the process's LPS.

The LPS represents the maximum possible privilege set that the process can have. The LPS is always a superset of the MPS. Any process can remove a privilege from its LPS, but there is no override mechanism to add a privilege to the LPS. A process cannot acquire privileges from an executable file if the privileges are not in the process's LPS.

Privileges are inherited by child processes just like the IDs associated with the process.

6.2.5.3 File Privilege Sets (PV.3)

There are three types of file privilege sets: *innate privilege* sets (IPS), *proxy privilege* sets (PPS), and *authorized privilege* sets (APS). File privilege sets have no effect on any file access or operations. They have meaning only for executable files and affect what privileges a process will have after executing the file.

The IPS is the set of privileges automatically granted to a process when the file is executed, subject to the limitation imposed by the process's LPS.

The PPS is the set of privileges that a process can keep after executing the file if the process had the privilege in its MPS prior to executing the file.

The APS is similar to the IPS except that the APS privileges are granted to the process only if the user associated with the process executing the file has at least one of the authorizations in the file's privileged authorization set (PAS). If a specific privilege is in both the IPS and the APS of a file, the existence in the IPS will take precedence and the privilege will be granted to the process regardless of the PAS of the file.

6.2.6 Authorizations (AZ)

Authorizations are the mechanism used to mark certain user accounts as being associated with special administrative roles, such as the information system security officer (ISSO), the system administrator (SA), or the system operator (SO). Authorizations can be used to enforce the two-man rule, such as adding new users to the system, where one user can add the account and another can set up the account security information.

Authorizations are used to limit access to privileged programs, such as the mount command and the shutdown command. This allows a distinction to be made between administrators and regular users and to divide administrative duties among different classes of administrators.

Authorizations are also used by trusted programs to provide different levels of functionality for different classes of users. For example, the *chmod* program will allow a user with ISSO authorization to change the permission bits of any file, but allow a regular user to change the permission bits only for his own files.

Authorizations are enforced even when superuser emulation mode is enabled. Thus some commands cannot be executed by user ID 0 even when SEM is enabled.

Authorizations may be used for the following:

1. to determine if a user (or a process running on the user's behalf) can run a program,
2. to determine what privileges a process can have or use while running a program, and
3. to determine what functionality is available to a user while running a specified program.

Cases 1 and 2 can be handled without any modification to the source or binary file. Case 3 requires that the source code of the program be modified to specifically check for authorization and to behave differently based on the authorization check.

6.2.6.1 Authorization Attributes (AZ.1)

An authorization consists of the following components:

- a unique name of at most 31 characters
- a unique numeric identifier in the range from 1 to 32767 inclusive
- a flag indicating if the authorization is a trusted path authorization
- a flag indicating if the authorization is available only in maintenance mode
- an optional authorization, called the *authorization role*, that a user must have before being allowed to be given the authorization
- up to four optional authorizations, called *dominating authorizations*, that include all of the rights associated with the authorization

6.2.6.2 Process Authorizations (AZ.2)

The kernel maintains a table of authorizations and associated user IDs, which it uses to check if a process has a particular authorization. The kernel authorization table (KAT) can only be modified by a process having the PV_AZ_ADMIN privilege in its EPS. A process can read the entire KAT only if it has the PV_AZ_READ privilege in its EPS. No privilege is required by a process to check if it is running as a user that has a particular authorization.

Each process has a limiting authorization set (LAS). The LAS is initialized for a user's session from a user account profile. When checking for an authorization in the KAT, the kernel will always return false if the authorization is not in the process's LAS. The LAS implementation allows up to 16 authorizations to be listed in the LAS. This set may either be the set of authorizations allowed for the process, or the opposite, the only authorizations this process may not acquire.

When a process is created via the *fork* system call, the new child process inherits the LAS of its parent.

Any process can remove an authorization from its LAS. A process can remove an authorization from another process's LAS only if the process making the change has the PV_AZ_ADMIN privilege in its EPS and only if it has MAC WRITE access to the process.

Because the authorization set associated with a process is a function of the process's RUID, the authorizations of a process may change dynamically, such as after a *setuid* system call. However, the LAS can be used to prevent a process from acquiring an authorization in this way.

6.2.6.3 File Authorization Sets (AZ.3)

Every file system object has two authorization sets: an *access authorization set* (AAS) and a *privileged authorization set* (PAS). File authorization sets are used only on executable files. When an object is created, the object's authorization set is set to NULL. A file may have up to four authorizations per set.

A process can change an authorization set of an object only if:

1. the process has ownership of the object,
2. the process has MAC WRITE access to the object,
3. the process has DAC WRITE access to the object, and
4. the process has the PV_AZ_SET privilege in its EPS.

If an executable file's AAS is not NULL, then a process can *exec* the file only if it has at least one of the authorizations in the AAS.

A file's PAS is used only if a process can otherwise successfully *exec* the file. The PAS is used only to give the *exec*'ing process access to the file's APS. The PAS does not restrict or enhance the ability of a process to *exec* the file.

6.2.6.4 Authorization Checks (AZ.4)

A process can inquire if it has a specified authorization and can get a list of all of its current authorizations without the need for any privilege.

An executable file's AAS is checked automatically during the *exec* system call.

If the system is not in Trusted Path mode, authorizations having the TP flag set will not be activated for the process.

If the system is not in maintenance mode, authorizations having the maintenance mode flag set will not be activated for the process.

6.2.6.5 Implementation (AZ.5)

A process's LAS is defined as a set of up to 16 authorizations. A process can be granted an authorization (based on its RUID) only if the authorization appears in the LAS.

6.2.7 Mandatory Access Control (MAC)

The TOE provides a MAC policy that enforces access to named objects listed in Table 4 based on sensitivity labels. A sensitivity label is an access control structure that enumerates the MAC access control properties for the object. Sensitivity labels exist for all subjects and named objects on the system. A sensitivity label contains a single classification and a set of zero or more categories. PitBull supports up to 32,767 hierarchical classifications and up to 1,024 categories.

The MAC policy enforced by the TOE can be described using the concept of sensitivity label dominance. A sensitivity label dominates another if the classification of the first equals or exceeds the classification of the second, and if every category in the second is included in the first. For a subject to read an object, the subject's sensitivity label must dominate the object's sensitivity label. PitBull uses a label encodings file to designate TOE labels and their dominance relationships that are used for MAC enforcement. For a subject to write an object, the object's sensitivity label must be equal to the subject's sensitivity label.

PitBull can be configured with an accreditation range. The accreditation range is defined by a System High and System Low sensitivity label. The System High label must dominate all data processed on the system. All data on the system must dominate the System Low sensitivity label.

Processes can use privileges to override MAC restrictions as defined in FDP_IFF.2(1).

The import and export of data PitBull is implemented through a trusted version of the *backup/restore* utilities. These utilities have been extended to handle labeling. The extensions are transparent to the user and, aside from the labeling extensions, the command functions in the standard fashion. A combination of privilege and authorization mechanisms protects the import/export system.

Import and Export of labeled data describes the system's ability to maintain security attributes when objects are imported and exported to and from the system using predefined security enforcing interfaces. Import and Export of

unlabeled data describes the system's ability to disregard security attributes, and the restrictions that are in place to maintain the system's integrity. PitBull enforces mandatory access control when exporting labeled and unlabeled data. Mandatory access control decision are based on the files sensitivity label and the configured device sensitivity label or label range. The subject exporting the data on behalf of a user must have appropriate privileges, or the user must have appropriate authorizations.

Labeled functionality is included on the TOE. Data can be sent to both hardware devices and to files. Labels are automatically included in backup headers. An unauthorized user cannot override the placing of labels with the data. Only an ISSO authorized administrator can export data without security labels. The TOE provides the tar command that can be used to export data, and does not preserve security information. The TOE requires that a manual change in the device state be performed before using a device to export data without security attributes. This action is auditable.

When writing to disks or tapes with *backup* sensitivity labels (SLs) are included with the data. ISSO authorization is required to use backup/restore to import or export unlabeled data from tapes or disks. When writing unlabeled data, the data receives the SL of the writing process. When importing unlabeled data, the data is assigned the SL of the importing process. However, if unlabeled data were written with a high-level process, the reading process must have an equal or higher level to read the data.

The evaluated TOE generates printer output using the Postscript Version 2.0 standard and PCL Version 5. The ISSO has the ability to specify the printable label assigned to the sensitivity label of a print job that is sent through the SystemV printing subsystem of AIX. Printed output is labeled with banner pages and security headers and footers. The banner pages contain the human readable sensitivity label.

Printers and mounted file systems are the devices that are available to users for importing/exporting **labeled** data only. PitBull places labels on the data being imported or exported via these devices. If an administrator changes the device such that it does not handle labeled data, the system is outside the evaluated configuration.

For remote access via NFS, as well as the other single level file systems cdrfs, udfs, and procfs, a single level policy is implemented: On the client side of NFS, as well as for the other single level file systems, the labels of the mount point are applied to the entire file system (precisely, the max SL and min SL for each file system object located within the single level file system is equal to the max SL and min SL of the mount point).

The following devices are available to users for importing/exporting **unlabeled** data only:

- floppy drive
- raw hard disk drive
- CD-ROM writeable
- serial port
- other devices

PitBull does not place labels on data being imported or exported via these devices. If an administrator changes the device such that it handles labeled data, the system is outside the evaluated configuration. Although, administrators can make these devices unavailable, this action does not change the device state from unlabeled to labeled.

6.2.8 Advanced Secure Networking (ASN)

The following devices can be configured for both labeled and unlabeled data.

- network

6.2.8.1 Network and interface rules (ASN.1)

Labeling of data is handled by ASN, which reads settings from rules loaded into the kernel from in configuration files. On initial installation, the rules are set to import and export labeled data. The rules can be changed with the netrule(1a) command. Such changes are not saved on power down. The asninit(1a) command saves changes between sessions.

The configuration files for ASN are /etc/asn/rules.host and /etc/asn/rules.int. These are in binary format. If /etc/asn/rules.int does not exist, the system will create a rules file from the ASCII text file /etc/asn/scripts/iniRules.txt by running asninit(1a) during the boot sequence.

The “change in device state” from labeled to unlabeled import and export of data is determined by the ASN rules. Therefore, using netrule(1a)or asninit(1a)to change the ASN rules constitutes the “manual change in device state” required for the evaluation.

The netrule(1a)and asninit(1a)commands generate audit events when rules are changed. This action satisfies the Security Target requirement for auditing the “change in device state”.

Exported data, even if labeled, may be read by a non-PitBull machine. For example, if an administrator uses backup to archive data onto a tape, then the tape can be read by an unmodified machine using restore, and all labels will be ignored, granting access to all data on the tape. Administrators should be aware that this is not a supported option.

ASN provides two sets of networking rules: network interface and host filtering. Both types of networking rules determine what processing occurs on a packet before its transmission or when it is received. These rules are an extension of the MAC policy because 1) they apply sensitivity labels to packets, and 2) they enforce MAC restrictions on packets according to those labels. PitBull enforces MAC between subjects and ASN named objects.

Network interface rules enforce packet label processing based on the physical network interface of the host. Host rules enforce packet label processing based on the source and destination IP addresses (with network masking allowed) of the packet, the source and destination ports of the request, and the protocol being used. Both types of rules provide several criteria for determining which packets to drop and which to pass.

Information flow is permitted only if the following sequential ordering relationships between security attributes hold:

a. if the IP address of the packet is equal to the IP address specified in the rule;

b. if the IP address of the packet is within the network mask specified for the rule;

c. if the direction of the packet flow corresponds to the direction of the rule (IN/OUT);

d. if the protocol of the packet is equal to the protocol specified in the rule;

e. if the port is within the range specified in the rule;

f. if the network interface of the packet is equal to the network interface specified in the rule;

g. if the IPSO labels are within the range defined by the rule, and the rule set to allow IPSO labels;

h. if the packet’s SL is within the minimum and maximum SL specified for the rule.

6.2.8.2 Internet Protocol Security Option (IPSO) (ASN.2)

The TOE supports the Revised Internet Protocol Security Option (RIPSO) specified in RFC 1108 and the Common Internet Protocol Security Option (CIPSO) as specified in FIPS PUB 188, which allows the transmission of labeled data over IP networks.

In order to preserve the labels for the transmitted data, IP datagrams are extended with IP options that provide for classification of the transmitted data. To translate system-specific sensitivity labels defined in /etc/security/LabelEncodings into the RFC1108-specified RIPSO labels, a translation table has to be created in (for example) /etc/asn/rfc1108 and its location to be supplied to *asninit*. CIPSO uses as security tag to indicate the rule used to construct the security data in the IPSO. ASN supports the tag types 1, 2 and 5 as specified in FIPS PUB 188.

The netrule(1) command (requiring the NETRULE authorization) is used to define ASN rules that specify the usage of CIPSO or RIPSO for network connections (see above).

6.2.9 Mandatory Integrity Control (MIC)

Mandatory integrity control is a system-enforced means of restricting access to and modification of objects based on the integrity of the object and the clearance of the user. While MAC is concerned with the sensitivity of an object, MIC is concerned with its trustworthiness.

The TOE enforces MIC using a system of labels. All named objects have integrity labels (TLs) to identify the integrity level of the object. Processes also have TLs. Process TLs indicate what level of information integrity the processes are allowed to access. The higher the TL, the more trustworthy the object or process. A process must be at least as trustworthy as an object in order to modify it. This means a process must have an TL equal to or exceeding that of the object. Thus TLs can be used to make files accessible for read only. For creating new objects in a file system, a process must be at least as trustworthy as the directory the object is to be created in.

MIC for read access is not enforced in the evaluated configuration.

6.2.9.1 MIC Labels (MIC.1)

All system objects, such as files, processes, windows, etc. have TLs. TLs are automatically placed on objects at the time of creation. All core dumps are considered objects on the system and are automatically labeled by the system. Objects on the system prior to PitBull installation receive TLs whenever accessed for the first time after PitBull installation.

Only processes with proper privileges and authorizations are able to change the TL of a file or process.

Unlike MAC, directories only have a single label for MIC. There is a special TL that can be put on a file or process, called NOTL. When NOTL is on an object or process, no MIC checks are performed on it. Only privileged users can set a TL to NOTL, or change a TL if it is currently NOTL.

6.2.10 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. Three general techniques are applied to meet this requirement in the AIX 5.3 system: explicit initialization of resources on initial allocation or creation, explicit clearing of resources on release or deallocation, and management of storage for resources that grow dynamically.

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 AIX 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.

6.2.10.1 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 both normal and large JFS2 file systems.

Object reuse is irrelevant for the CD-ROM File System (CDRFS) and the Universal Data Standard file system (UDFS) because they are read-only file systems, making it impossible for a user to read residual data left by a previous user. File systems on other media (tapes, diskettes) are irrelevant because of warnings in the Security Guide not to mount file systems on these devices.

For this analysis, the term FSO refers not only to named file system objects (files, directories, device special files, named pipes, and UNIX domain sockets) but also to unnamed abstractions that use file system storage (symbolic links and unnamed pipes). All of these, except unnamed pipes, device special files and UNIX domain sockets, have a directory entry that contains 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 their contents are fully specified at creation time.

6.2.10.1.1 Object Reuse: Files

Storage for files is allocated automatically in pages as a file grows. These pages are cleared before they become accessible, within the file. However, when a file is deleted the space holding the data from the file, both in memory and on disk, is not cleared. This data will persist until the space is assigned to another file, when it will be cleared. These internal fragments of deleted files are protected by the kernel to prevent accessing of deleted data.

If data is read before it is written, it will read only as zeroes. Reads terminate when the end-of-file (EOF) is detected. It is possible to seek past the EOF, but any reads will return zeros. File writes may cause the file to grow, thus overwriting any residual data and moving the EOF. If the file pointer is advanced past the EOF and then written, this leaves a hole in the file that will subsequently be read as zeroes.

6.2.10.1.2 Object Reuse: Directories and Directory Entries

In part, object reuse for directories is handled as for ordinary files: pages allocated are always cleared before being incorporated into the directory. When a directory is first created, it is explicitly initialized to have the entries "." and "..", but the remainder of the directory's storage is cleared.

Individual directory entries are manipulated as distinct resources, such as when referencing file system objects, and as part of the directory, such as when reading the entire directory itself. When a directory entry is removed or renamed the space occupied by that directory entry is either combined with the previous entry as free space or else the inode number of the entry is set to zero when the entry occurs on a 512 byte boundary.

When a directory entry does not occur on a 512-byte boundary, the size of the preceding directory entry is incremented by the size of the directory entry which has been removed. The space in a directory entry in excess of that which is needed to store the necessary information may be allocated when a directory entry is to be created. The fields of the directory entry remain unchanged.

When a directory entry occurs on a 512-byte boundary, the inode number is set to zero to indicate that this entry is now available for re-use. All other fields of the directory entry remain unchanged.

The directory entry is no longer visible to interfaces which perform file name operations and may only be seen when the entire directory is examined and the process has read access to the directory.

6.2.10.1.3 Object Reuse: Symbolic Links

Symbolic links have their contents (the link pathname) fully specified at creation time, and the readlink operation returns only the string specified at creation time, not the entire contents of the block it occupies.

6.2.10.1.4 Object Reuse: Device Special Files

All device special files are initialized to a known state on first open and never grow. Device special files refer to actual hardware or else to virtualized objects. There are no file system blocks, unless the device references a file system (in which case the mechanism for object reuse of file system objects apply). Nor is there memory, unless the device is associated with memory (in which case the object reuse mechanisms for memory objects apply).

6.2.10.1.5 Object Reuse: Named Pipes

FIFOs are created empty. Buffers are allocated to contain data written to a pipe, but the read and write pointers are managed to ensure that only data that was written to the pipe can ever be read from it.

6.2.10.1.6 Object Reuse: Unnamed Pipes

Unnamed pipes are created empty. Buffers are allocated to contain data written to a pipe, but the read and write pointers are managed to ensure that only data that was written to the pipe can ever be read from it.

6.2.10.1.7 Object Reuse: Socket Special File (UNIX Domain)

UNIX domain sockets have no contents; they are fully initialized at creation time.

6.2.10.2 Object Reuse: IPC Objects (OR.2)

AIX 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 256 MBytes, semaphore is one bit), and so there is no way to grow the object beyond its initial size.

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

6.2.10.3 Object Reuse: Queuing System Objects (OR.3)

6.2.10.3.1 Object Reuse: Batch Queue Entries

cron and at jobs are defined in batch files, which are subject to the object reuse protections specified for files as described previously.

6.2.10.4 Object Reuse: Miscellaneous Objects (OR.4)

6.2.10.4.1 Object Reuse: Process

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.

The AIX 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. When a process requests more memory, the memory is explicitly cleared before the process can gain access to it.

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. 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.

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. When a process execs a new program, the text segment is replaced entirely.

6.2.10.5 Object Reuse: Hard disk drives (OR.5)

For SCSI disks that do not participate as "pdisks" in RAID arrays, the diagnostic subsystem (cf. section 6.2.12.7) offers a "Hard File Erase Disk" functionality to administrators of the TOE. SCSI disks which are part of a pdisk must be detached from the pdisk for erasure.

This option can be used to overwrite (remove) all data stored in currently user-accessible blocks of the disk. The Erase Disk option writes one or more user-specifiable patterns to the disk.

The administrator can specify the number (0-3) of patterns to be written to the hard disk. The patterns are written serially; that is, the first pattern is written to all blocks. Then the next pattern is written to all blocks, overlaying the previous pattern. Also, a random pattern can be written.

Please note that there are two abstraction layers in the underlying environment involved that restrict the TOE to the deletion of user-accessible blocks on those hard disk drives only: Firmware of SCSI hard disk drives and firmware of SCSI disk controllers may remap "bad blocks" containing user or TSF data to healthy blocks on the physical hard disk drive and maintain a pool of unallocated blocks for this purpose. The TOE is not able to (and does not claim to) overwrite such blocks, since it is using the generic SCSI interfaces to access the hard disk drive. Since the hard disk drive stays within the TSC it is ensured that users of the TOE, accessing the drive via the TOE-provided interfaces, won't be able to recover any residual information on it.

6.2.11 Security Management (SM)

This section describes the functions for the management of security attributes that exist within AIX 5.3.

6.2.11.1 Roles (SM.1)

PitBull provides three distinct administrative roles, Information System Security Officer (ISSO), System Administrator (SA), and System Operator (SO), and provides the ability to define other site-based roles. The TOE associates authorization sets with a role, and ensures that the authorization required to assume a role are satisfied before allowing operations associated with the role to be performed.

In the evaluated configuration the administrator is usually not root, but one of the ISSO, SA and SO roles. Root itself will not be used as a user ID where a user can directly log in to or for administration.

Users that have the appropriate authorizations associated with their account may run administrative programs associated with those authorizations/roles.

6.2.11.1.1 Normal Users

Normal users can not perform actions that require administrator privileges. They can only execute those setuid root programs and privileged programs they have access to (either via DAC or authorizations). 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(1).

6.2.11.1.2 VIOS Roles

All VIOS roles are authorized administrative roles. VIOS doesn't support the concept of normal users. VIOS defines the following roles:

- Prime Administrator (a.k.a. padmin) – This role can execute every command provided by the VIOS command-line interface including the user ID commands and security commands. This role is limited to a single user ID: *padmin*. This user ID is defined in the installation image and no other users can be a Prime Administrator.
- System Administrator– This role can execute every command provided by the VIOS command-line interface except for the security commands and user ID commands (exception: they can change their own passwords). System Administrator user accounts do not exist until the Prime Administrator creates one or more.
- Development Engineer (DE) – This role is used only by IBM personnel to debug problems and run diagnostics. Development Engineer user accounts do not exist until the Prime Administrator creates one or more.
- Service Representative (SR) – This role allows a service representative to run commands that are required to service the system (shutdown, restart, update system microcode, configure/unconfigure devices, certify, format, etc.). Service Representative user accounts do not exist until the Prime Administrator creates one or more.

This function contributes to satisfy the security requirement FMT_SMR.1(2).

6.2.11.2 Audit Configuration and Management (SM.2)

Audit control consists of the files used to maintain the configuration of the audit subsystem and a description of the AUDIT command and its associated parameters

Table 10: Audit Control Files. The audit control files maintain the configuration for the auditing subsystem.

Audit Control File	Description
/etc/security/audit/config	Defines whether bin mode auditing is enabled, the names of the files used to store audit data and the names of the available classes. Also defines the audit classes, i.e., for each audit class the audit events belonging to the class are defined
/etc/security/audit/events	Defines audit events to be used on the system. An event needs to be defined in this file to be formatted correctly.
/etc/security/audit/objects	Contains a list of the objects whose access will be audited
/etc/security/audit/bincmds	Contains the post-processing command or commands for bin mode auditing
/etc/security/user	Contains a record for each user which specifies which classes will apply to the user account

There are two different types of audit event selection: per-user and per-object. Per-user auditing allows the administrator to specify specific classes of audit events that will be recorded for that user. Each process stores a copy of the audit classes that apply to that user as part of the process table. An audit class is a subset of the total number of audit events available and is defined in the file */etc/security/audit/config*.

Per-object auditing allows the administrator to specify file system objects that will be audited. This is defined in the file */etc/security/audit/objects*. There for individual objects the audit event for the access modes that one wants to be audited is defined.

These objects can be audited based on accesses of a specified mode (read/write/execute) and record the result of the access attempt (success/failure).

The *audit* command is used to start and stop the auditing subsystem, to temporarily switch the auditing subsystem on or off, and to query the audit subsystem for the current audit parameters.

The *audit* command is started from the host's rc initialization script, as stated in the Security Guide.

The on and off parameters of the *audit* command enable and disable audit, without modifying the current configuration that is stored in the kernel. The on parameter can have an additional parameter, panic, which causes the system to shut down if bin data collection is enabled and records cannot be written to one of the bin files. The bin mode panic option can also be specified in */etc/security/audit/config*.

When the *audit* command is issued with the shutdown parameter, the collection of audit records is halted, and all audit configuration information is flushed from the kernel's tables. All audit records are flushed from the kernel's buffers and processed. The collection of audit data is halted until the next audit start command is entered.

When the *audit* command is issued with the start parameter, the following events occur:

- the */etc/security/audit/config* file is read
- the */etc/security/audit/objects* files is read and the objects that will be audited based on access are written into kernel tables
- the audit class definitions are written into kernel tables from */etc/security/audit/config*
- the auditbin daemon is started, depending on the options in */etc/security/audit/config*
- auditing is enabled for users specified in the user's stanza of the */etc/security/audit/config* file
- auditing is turned on, with panic mode enabled or turned off, depending on what mode is specified in */etc/security/audit/config*

To access the audit functions the appropriate PV_AU privileges are required. The AUDIT authorization is needed for reading audit records.

The events that are audited can be selected on a per user basis, per event basis and per object basis using the configuration files described above.

A description of the structure of those files and the syntax of the entries can be found in *AIX 5.3 Files Reference* document.

6.2.11.3 Access Control Configuration and Management (SM.3)

Discretionary access control to objects is defined by the permission bits and by the Access Control Lists (for those objects that have access control lists associated with them) or by NFSv4 ACLs. 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 system administrator can define and modify those default values.

Permissions can be changed by the object owner and the system administrator. When an object is created the creator is the object owner. Object ownership can be transferred except for TCP ports, where the owner always remains the system administrator. In the case of IPC objects, the creator will always have the same right as the owner, even when the ownership has been transferred.

For MAC, ASN and MIC, sensitivity and integrity labels are assigned to all objects and users, whereas users are assigned a specific range of levels (clearance) within which they can operate. TCB files are identified by a specific file security flag that can be set by authorized administrators. Authorizations can be granted to users by authorized administrators.

NFSv4 ACLs provide a mechanism which allows the object owner to give others the ability to modify the entries within the ACL. Directory NFSv4 ACLs can include entries that are inherited by child objects.

For VIOS, both the VIOS SCSI discretionary access control and the VIOS Ethernet discretionary access control are managed by the system administrators. VIOS provides an administrative interface for managing these functions. VIOS SCSI device drivers acting on behalf of the LPAR partitions are not allowed to access a logical or physical volume until the mapping is created in VIOS. A VIOS Ethernet device driver acting on behalf of a group of LPAR partitions sharing a virtual network cannot access a VOS Ethernet adapter device driver and vise versa until a mapping is created in VIOS.

6.2.11.4 Management of User, Group and Authentication Data (SM.4)

6.2.11.4.1 Creating new Users

An administrator (role SA) can create a new user and assigns a unique user ID to this user. The initial password and other security attributes have to be defined by the ISSO using various utilities. The new user will be disabled until the initial password is set.

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

- Lock attribute (i.e., temporarily locking a user account)
- Administrative status of the user
- List of audit classes for the user (AIX only)
- List of groups the user belongs to
- Home directory for this user
- Number of consecutive unsuccessful login attempts allowed before the user account is locked
- Password parameter including the maximum and minimum age of a password, minimum length, difference to the old password, etc.

Those attributes are stored in the following files:

- */etc/passwd*
- */etc/security/azdb*
- */etc/security/clear*
- */etc/security/las.*
- */etc/security/passwd*
- */etc/security/user*
- */etc/security/audit/config*

6.2.11.4.2 Modification of user attributes

User attributes can be modified by the system administrator (ISSO). Modifications of user attributes require the modification of the administration database that contains the user attributes (mainly */etc/security/user*).

6.2.11.4.3 Management of Authentication Data

The system administrator (ISSO) has the capability to define rules and restrictions for passwords used to authenticate users. The parameters available are:

minage	Minimum number of weeks that must pass before a password can be changed.
maxage	Maximum number of weeks that can pass before a password must be changed.
maxexpired	Maximum number of weeks beyond maxage that a password can be changed before administrative action is required to change the password. (Root is exempt.)
minalpha	Minimum number of alphabetic characters the new password must contain.
minother	Minimum number of nonalphabetic characters the new password must contain. (Other characters are any ASCII printable characters that are nonalphabetic and are not national language code points).
minlen	Minimum number of characters the new password must contain.
maxrepeats	Maximum number of times a character can be used in the new password.
mindiff	Minimum number of characters in the new password that must be different from the characters in the old password.

histexpire	Number of weeks that a user is unable to reuse a password.
histsize	Number of previous passwords that cannot be reused.
dictionlist	List of dictionary files checked when a password is changed. Dictionary files contain passwords that are not allowable.

Users are also allowed to change their own password using the *passwd* command. The password restrictions defined by the system administrator apply.

VIOS supports the following passwords parameters only: maxage, maxexpired, minother, minlen, maxrepeats, histexpire, and histsize.

6.2.11.5 Time Management (SM.5)

AIX 5.3 provides the standard Unix functions to manage the system clock. The time can be set or modified by an administrator (ISSO). Modifications to the system time are audited (if configured) allowing a system administrator to extract the differences between the "old" and "new" value of the system clock. The value of the system clock can not be manipulated by normal users.

6.2.12 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 of AIX 5.3. 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, MAC, MIC, and TCB controls and process isolation mechanisms. In general, files and directories containing internal TSF data (e.g., audit files, batch job queues) are also protected from reading by access control permissions.

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

The boot image for each system is adequately protected. A description of the boot logical volume can be found in section 5.3.16, Initialization and Shutdown.

6.2.12.1 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.

Resources managed by the kernel software can only be manipulated while running in kernel mode.

Processes run in user mode and can call functions of the kernel only as the result of an exception or interrupt. 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. Other trusted process interfaces are started during system initialization and use well defined protocol or file system mechanisms to receive requests.

Some system calls or parameters of system calls are reserved are reserved for trusted processes. When called the kernel checks that the calling process runs with the appropriate privileges.

When configured by an administrator, the kernel ensures that code residing on the stack of selected processes cannot be executed by the processes.

The TOE implements the TSP through a reference monitor. The kernel reference monitor (KRM) is a single C-language function that is called any time the security policy may need to be enforced. The KRM accepts as arguments all of the security attributes of the subject and object associated with the security check, along with an indication of what check or

checks and auditing need to be performed. The KRM itself consists of many code modules that are called to handle the appropriate check or checks that are required. The KRM is used to enforce the security policy in the following instances:

- Systems calls
- File system creation, deletion, and access events
- Interprocess communication, including signals
- Network packet checks

The KRM handles all of the following kernel security mediation events:

- Discretionary access control (DAC) checks
- Mandatory access control (MAC) checks
- Advanced Secure Networking (ASN) rules
- Mandatory integrity control (MIC) checks
- Trusted computing base (TCB) checks
- System security flag (SSF) checks
- File security flag (FSF) checks
- Privilege (PV) checks
- Authorization (AZ) checks
- Auditing (AUD) for all of the above

The KRM accepts as arguments the following:

- subject security attributes
- action to be performed by the reference monitor
- security attributes and object type for up to 4 objects
- flag indicating if auditing should be done
- pointer to the system-wide security settings

Although the KRM can check MAC, MIC, DAC, TCB, FSF, SSF, PV, AZ, and AUD components, not all actions require all checks. When multiple checks are required, they are performed in the following order:

- MAC (with SSF, FSF, PV and AUD as needed)
- MIC (with SSF, FSF, PV and AUD as needed)
- FSF (with PV and AUD as needed)
- TCB (with SSF, PV and AUD as needed)
- DAC (with PV and AUD as needed)
- PV (with SSF, FSF, and AUD as needed)
- AZ (with SSF, FSF, PV, and AUD as needed)

System security flags (SSF), also known as kernel security flags, support the enforcement of the TSP. The evaluated configuration mandates the following settings for SSF in order to ensure that the TSP be enforced accurately:

Table 11: System/kernel security flags for the evaluated configuration.

SYSTEM SECURITY FLAG	OPERATIONAL MODE	CONFIGURATION/ MAINTENANCE MODE
ASN [A]	ENABLED	ENABLED
AZROOT [Z]	**DISABLED**	**ENABLED**
MLS [B]	ENABLED	ENABLED
CAPABILITIES [M]	DISABLED	DISABLED
Foundation Suite [K]	DISABLED	DISABLED

SL_ENFORCEMENT [J]	ENABLED	ENABLED
PRIV [P]	ENABLED	ENABLED
PVROOT [V]	ENABLED	ENABLED
STRPUSH [U]	ENABLED	ENABLED
SU_EMUL [R]	DISABLED	DISABLED
TCB [E]	ENABLED	ENABLED
TDE [D]	DISABLED	DISABLED
TL_WRITE_ENFORCEMENT [G]	ENABLED	ENABLED
TL_READ_ENFORCEMENT [H]	DISABLED	DISABLED
XPRIVS [T]	DISABLED	DISABLED
XSL [X]	DISABLED	DISABLED
TLIBPATH_OFF [S]	DISABLED	DISABLED

6.2.12.2 Kernel (TP.2)

The AIX 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).

The kernel is shared by all processes, and manages system wide shared resources. It presents the primary programming interface for AIX 5.3 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 low-level I/O system, and the kernel extensions like implementing for example network protocols (IP, TCP, UDP, and NFS).

6.2.12.3 Kernel Extensions (TP.3)

Kernel extensions are dynamically loaded code modules that add function to the kernel. They include device drivers, virtual file systems (e.g., CDRFS), inter process communication methods (e.g., named pipes), networking protocols, and other supporting services. Kernel extensions can be loaded only at system boot in the evaluated configuration.

Kernel extensions run with kernel privilege, similarly to kprocs. However, extensions differ from kprocs in that the kernel does not schedule them. Instead, kernel extensions are invoked from user processes by system calls, or internal calls within the kernel, or started to handle external events such as interrupts.

Kernel extensions run entirely within the kernel protection domain. An extension may export system calls in addition to those exported by the base AIX kernel. User-domain code can only access these extensions through the exported system calls, or indirectly via the system calls exported by the base kernel.

Device drivers are kernel extensions that manage specific peripheral devices used by the operating system. Device drivers shield the operating system from device-specific details and provide a common I/O model for user programs to access the associated devices. For example, a user process calls read to read data, write to write data, and ioctl to perform I/O control functions.

6.2.12.4 Trusted Processes (TP.4)

Trusted processes in AIX are processes running in user mode but with privileges. Some high-level TSF functions are performed by trusted processes particularly those providing distributed services.

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 administrators or during system initialization).

Trusted processes have all the kernel interfaces for which they have the appropriate privilege 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, audit data management and reduction, some network operations, system initialization, and system administration.

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

Any program executed with DAC override privileges has the ability to perform the actions of a trusted process. It is therefore important that a site operating AIX strictly controls those programs and prohibits that those programs are modified or that programs from untrusted sources are executed with root privileges.

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.

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.

6.2.12.5 TSF Databases (TP.5)

Table 12identifies the primary TSF databases used in AIX 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 administrators. None of these databases shall be modifiable by a user other than a system administrator with the appropriate authorization.

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.

VIOS, which uses only file-based I&A, uses the same TSF database files as AIX.

When the NFSv4 client and server map user/group string names to UIDs/GIDs, they use the local OS authentication mechanism to make the mapping; thus, they will use the file-based I&A mechanism.

Table 12: Administrative Databases. This table lists other administrative files used to configure the TSF.

Database	Purpose
/etc/asn/rules.host and /etc/asn/rules.int	Configuration files for network port protection and labels
/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

Database	Purpose
	server.
/etc/IandA	Specifies optional extensions to the login mechanism.
/etc/inittab	Controls the system startup by running the appropriate command scripts and SRC invocations
/etc/passwd	Stores user names, UIDs, primary GID, home directories for all system users.
/etc/security/audit/bincmds	Specifies the pipeline of commands to be performed by the auditbin daemon.
/etc/security/audit/config	Specifies who and what is going to be audited, where the bin audit data will reside, and how auditing will be performed.
/etc/security/audit/events	Defines all of the audit events that are recognized by the system and the form of their tail data.
/etc/security/audit/objects	Specifies file system objects whose access is to be audited along with for what access modes it will be done.
/etc/security/azdb	Authorizations database
/etc/security/clear	Contains default SL, min SCL, max SCL, default TL, min TCL and max TCL for all users on the system.
/etc/security/device_levels	Sensitivity labels for devices
/etc/security/integrity/*	Integrity checking database
/etc/security/LabelEncodings	label mappings
/etc/security/las	User authorization profiles used to grant authorizations to users at login time
/etc/security/lastlog	Stores time/date of last successful and unsuccessful login attempts for each user. Stores the number of unsuccessful login attempts since the last successful one.
/etc/security/libpath.txt	Contains the trusted library path
/etc/security/passwd	Defines user passwords in one-way encrypted form, plus additional characteristics including previous passwords, password quality parameters.
/etc/security/secconfig	Site security settings
/etc/security/secconfig.configuration	Kernel security flags to be loaded at boot time
/etc/security/ttys	Specifies permitted sensitivity label ranges for tty type devices
/etc/security/user	Defines supplementary data about users, including audit status, required password characteristics, access to su command.

These tables are not functions but they are part of the management of the TSF

6.2.12.6 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.

6.2.12.7 Diagnosis (TP.7)

AIX 5.3 provides a diagnosis program that can be used to check the correct operation of the underlying hardware of the system. This program can be executed by administrators or by hardware maintenance personnel. Results of the diagnosis program are stored in the diagnostic error log file, which can be protected by the discretionary access control functions of AIX against access by normal users.

6.2.12.8 Integrity Checks (TP.8)

For AIX only, the directory /etc/security/integrity contains the file system integrity database files. Each file is specific to a package and defines security attributes for the metadata of critical system files in the package, such as the DAC permission bits, owner, group, sensitivity labels, integrity labels, privileges, ACLs, authorizations and file security flags.

The command chkintegrity verifies the attributes of file system objects by comparing them to the stored values in the integrity database in /etc/security/integrity. If the attributes of the object do not match what is recorded in the integrity database, chkintegrity reports both the actual attributes and what they should be. Chkintegrity is run during startup of the system and can be manually executed by an authorized administrator at any time.

6.2.12.9 File security flags (TP.9)

In AIX only, file security flags (FSF) are used to mark files with various types of information which are then evaluated as part of the checks implemented in the reference monitor. The file security flags supported in the evaluated configuration are identified and explained in the following table:

Table 13: File Security Flags (FSF)

FSF	**Semantics of the flag being enabled/set**
FSF_APPEND	If this FSF is set, a file can only be appended to and not altered otherwise in operational mode. In configuration mode, this can be overridden by the PV_TCB privilege.
FSF_AUDIT	Marks a file as being part of the audit subsystem and allows reading only if a process has the PV_AU_READ privilege and writing only with the PV_AU_WRITE privilege. (See also AU.7)
FSF_EPS	For executables, the Effective Privilege Set of a process will be set to its new Maximum Privilege Set instead of being set to NULL.
FSF_IL_NF_OBJ	Not used in the evaluated configuration.
FSF_IL_NF_PROC	Not used in the evaluated configuration.
FSF_MAC_EXEMPT	A process with the PV_MAC_OVRRD privilege will ignore MAC restrictions when attempting to access the file system object.
FSF_MONITOR	When enabled, the audit subsystem will record accesses to the file regardless of other audit classes in the audit mask of a process.
FSF_PDIR	Identifies a partitioned directory.
FSF_PSDIR	Identifies a partitioned subdirectory.
FSF_PSSDIR	Identifies a partitioned sub-subdirectory.
FSF_TCB	The object is marked as part of the TCB. In the evaluated configuration, this flag can only be changed when the system is in configuration mode. To set or unset this flag, a process must have the PV_TCB privilege.
FSF_TCBPROC	For executables, a process will be marked as a TCB process and will only be able to link shared libraries that have the FSF_TCB flag set. In the evaluated configuration, this flag can only be changed when the system is in configuration mode and with the PV_TCB privilege.

6.3 Supporting functions not part of the TSF

6.3.1 System Management Tools

The evaluated configuration of AIX 5.3 provides the “System Management Interface Tool” (SMIT) for administration. This tool provides a more convenient way for an administrator to perform the administration activities. The web based administration tool WebSM is not included in the evaluated configuration and shall therefore not be used for system administration when operating the evaluated configuration of AIX 5.3.

SMIT itself is just a front-end tool which provides the administrator with a nice interface. SMIT generates scripts that use the system management commands provided by AIX. The administrator can review the shell scripts before they are executed by hitting the function key F6 before he executes the script.

In addition the administrator can use the commands provided by AIX for system management activities. Those commands are also 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 administrator 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. SMIT generates a script with commands that the administrator can check before it is executed. Therefore SMIT itself is not seen as part of the TSF. The commands themselves are also bound by the restrictions imposed by the system call interface and the access rights to the files in the administrative database.

6.3.2 User Processes

The AIX 5.3 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 are met by AIX 5.3.

Table 14: Mapping Assurance Requirements to Documentation

Assurance Component	Documentation describing how the requirements are met
ACM_AUT.1	CMVC is the tool used for configuration management of IBM AIX source code, documentation test plans and test cases. The PitBull extensions and the modified AIX source code are managed with ACMS, the Argus configuration management system
ACM_CAP.4	See above
ACM_SCP.2	IBM has a problem tracking procedure in place that covers all aspects of ACM_SCP.2. This description is included in the documentation of the configuration management and the software development procedures. Argus uses bugzilla to track problems. ACMS is capable of labeling changes based on problem reports.
ADO_DEL.2	Delivery procedures are documented in a separate document.
ADO_IGS.1	Installation, generation and start-up procedures are described as part of the guidance provided with the TOE.
ADV_FSP.2	A functional specification is provided.
ADV_HLD.2	A high-level design is provided.
ADV_IMP.1	Source code samples will be provided as requested for the evaluation.
ADV_LLD.1	Low-level design documentation is provided for all subsystems that implement TSF.
ADV_RCR.1	The correspondence information will be provided.
ADV_SPM.1	A separate document describing the Security Policy Model is provided.
AGD_ADM.1	Administrator guidance is provided with the TOE.
AGD_USR.1	User guidance is provided with the TOE.
ALC_DVS.1	The development security procedures are documented.

Assurance Component	Documentation describing how the requirements are met
ALC_FLR.1	A defect handling procedure is in place.
ALC_LCD.1	The life cycle definition is described in separate documents.
ALC_TAT.1	See above
ATE_COV.2	Detailed test plans and a test coverage and depth analysis are produced to test the TOE.
ATE_DPT.1	See above
ATE_FUN.1	Testing will be performed on different platforms that are defined in this Security Target. Test results will be documented.
ATE_IND.2	The evaluation facility will perform and document independent tests.
AVA_MSU.2	The misuse analysis will be provided.
AVA_SOF.1	The Strength of Function Analysis will be provided .
AVA_VLA.2	A vulnerability analysis will be provided.

6.5 TOE Security Functions requiring a Strength of Function

Please refer to section 8.4.3 for a discussion of SOF-rated mechanisms.

7 Protection Profile Claims

7.1 PP Reference

This Security Target claims conformance with the "Labeled Security Protection Profile (LSPP)", Version 1.b, 8 October 1999. This Protection Profile was developed by the "Information System Security Organization" 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".

7.2 PP Tailoring

Please refer to Table 2 and Table 6 for an identification of SFRs that have been added in addition to the ones derived from LSPP, and for an identification of the operations performed on the SFRs derived from the LSPP.

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.

Section 8.1 provides rationale for the augmentation of the security problem definition in chapter 3. Please note that P. AUTHORIZATION has been renamed in this ST to P.DATAFLOW to avoid confusion with the TOE's authorization mechanism.

The following security objectives for the TOE have been added:

- O.ERASE
- O.MANDATORY_INTEGRITY
- O.NETWORK_ACCESS
- O.TCB_ACCESS
- O.STACK
- O.VIOS

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

- OE.ADMIN
- OE.INFO_PROTECT
- OE.MAINTENANCE
- OE.RECOVER
- OE.SOFTWARE_IN
- OE.SERIAL_LOGIN
- OE.PROTECT
- OE.HW_SEP
- OE.LPAR

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.1. 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.1 which has been added to the assurance requirements defined in the LSPP 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.

The rationale is based on the rationale already provided in the Labeled Security Protection Profile. In accordance with the "Guide for the production of protection profiles and security targets" [GUIDE], only those aspects are discussed that are additional to the rationale provided in [LSPP].

8.1 Rationale for the augmentation of the security problem definition

The LSPP does not lists threats but only addresses the following policies:

- P.ACCOUNTABILITY
- P. CLASSIFICATION (renamed in this ST to P.DATAFLOW to avoid confusion with the TOE's authorization mechanism)
- P.AUTHORIZED_USERS
- P.NEED_TO_KNOW

Additional policies have been added to this Security Target:

- P.ERASE
- P.INTEGRITY
- P.STATIC
- P.TCBINTEGRITY
- P.DIST_USERS

[LSPP] also lists the following assumptions:

- A.CLEARANCE
- A.CONNECT
- A.COOP
- A.LOCATE
- A.MANAGE
- A.NO_EVIL_ADM
- A.PEER
- A.PROTECT
- A.SENSITIVITY

Three assumptions have been added:

- A.NET_COMP
- A.UTRAIN
- A.UTRUST

The Security Target has added a list of threats to be countered by the TOE or the TOE environment.

Threats countered by the TOE are:

- T.UAACCESS
- T.UAACTION
- T.UAUSER

- T.VIOS

Threats countered by the TOE environment are:

- TE.COR_FILE
- TE.HW_SEP
- TE.HWMF
- TE.LPAR

8.1.1 Rationale for additional Organizational Security Policies

Additional OSPs have been added to reflect the fact that the TOE runs on machines that offer dynamic partitioned environments (P.STATIC), enforces a mandatory integrity control (P.INTEGRITY), implements a trusted computing base mechanism (P.TCBINTEGRITY) and implements a policy for HDD erasure (P.ERASE).

8.1.2 Rationale for additional Assumptions

A.UTRAIN has been added as an assumption because it was also mentioned in the ITSEC Security Target for AIX. A.UTRAIN makes the assumption that users are trained well enough to use the security functionality provided by the system appropriately. This addresses the aspect of access control, where a user is responsible to manage access control rights for file system and IPC objects he owns. Users that need to protect their assets from unauthorized access by other authorized users of the system need to understand the implications of managing the access rights to file system objects and IPC objects they own.

User need also to be trained in the protection of their authentication data in the TOE environment.

A.UTRUST has been added as an assumption because it was also mentioned in the ITSEC Security Target for AIX. A.UTRUST makes the assumption that users are trusted some tasks or group of tasks within a secure IT environment by exercising complete control over their data. This also addresses the aspect of access control where user are trusted to use the access control mechanism provided by the TOE appropriately. Users should not blame the system for the loss of integrity and / or confidentiality of data they own when they don't use the access control mechanism provided by AIX appropriately.

A.NET_COMP has been added as an assumption to reflect the fact that the TOE is used in a distributed environment where network components are involved in the communication. It is assumed that those network components do not modify data transmitted over the network. This assumption is necessary, since some TSF rely on the correctness of data transmitted over the network to remote external systems.

8.1.3 Rationale for the inclusion of Threats

The Protection Profile has derived all security objectives from the organizational security policies listed in the Protection Profile. The authors of this Security Target decided to include also threats the TOE is going to counter as well as threats that need to be countered in the environment. Since also all the policies and assumptions defined in the Protection Profile have been included in the Security Target, the inclusion of threats is not a violation of the conformance claim to the Security Target.

8.2 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.

8.2.1 Security Objectives Coverage

Table 15: Mapping Objectives to threats , assumptions and policies

Objective	Threat / Policy
O.AUDITING	T.UAACTION, T.UAUSER, P.ACCOUNTABILITY
O.AUTHORIZATION	T.UAUSER, P.AUTHORIZED_USERS

Objective	Threat / Policy
O.DISCRETIONARY_ACCESS	T.UAACCESS, P.NEED_TO_KNOW
O.ENFORCEMENT	P.AUTHORIZED_USERS, P.NEED_TO_KNOW, P.ACCOUNTABILITY, P.DATAFLOW, P.INTEGRITY, P.DIST_USERS
O.ERASE	P.ERASE
O.MANAGE	P.AUTHORIZED_USERS, P.NEED_TO_KNOW, P.ACCOUNTABILITY, P.DATAFLOW, P.INTEGRITY, T.UAUSER, T.UAACTION
O.MANDATORY_ACCESS	P.DATAFLOW
O.MANDATORY_INTEGRITY	P.INTEGRITY
O.NETWORK_ACCESS	P.DATAFLOW
O.RESIDUAL_INFORMATION	P.DATAFLOW, P.NEED_TO_KNOW, T.UAACCESS
O.STACK	T.UAUSER, T.UAACCESS, P.AUTHORIZED_USERS
O.TCB_ACCESS	T.UAACCESS, P.TCBINTEGRITY
O.VIOS	T.VIOS

Table 16: Mapping objectives for the environment to threats, assumptions and policies

Environment Objective	Threat / Assumption / Policy
OE.ADMIN	A.MANAGE, A. NO_EVIL_ADM
OE.CREDEN	A.COOP
OE.HW_SEP	TE.HW_SEP
OE.INFO_PROTECT	TE.COR_FILE, A.PROTECT, A.UTRAIN, A.UTRUST, A.CLEARANCE, A.SENSITIVITY
OE.INSTALL	TE.COR_FILE, A.MANAGE, A. NO_EVIL_ADM, A.PEER, A.NET_COMP, P.STATIC
OE.LPAR	TE.LPAR
OE.MAINTENANCE	TE.HWMF
OE.PHYSICAL	A.LOCATE, A.PROTECT, A.CONNECT
OE.PROTECT	TE.COR_FILE, A.NET_COMP, A.CONNECT
OE.RECOVER	TE.HWMF, TE.COR_FILE
OE.SERIAL_LOGIN	A.CONNECT
OE.SOFTWARE_IN	P.NEED_TO_KNOW

Table 17: Mapping threats to objectives

Threat	Objective
T.UAACCESS	O.DISCRETIONARY_ACCESS, O.RESIDUAL_INFORMATION, O.TCB_ACCESS, O.STACK
T.UAACTION	O.AUDITING, O.MANAGE
T.UAUSER	O.AUDITING, O.AUTHORIZATION, O.MANAGE, O.STACK
T.VIOS	O.VIOS
TE.COR_FILE	OE.PROTECT, OE.INSTALL, OE.INFO_PROTECT, OE.RECOVER
TE.HW_SEP	OE.HW_SEP

TE.HWMF	OE.MAINTENANCE, OE.RECOVER
TE.LPAR	OE.LPAR

Table 18: Mapping Assumptions to Objectives

Assumption	Objective
A.NO_EVIL_ADM	OE.ADMIN, OE.INSTALL
A.CLEARANCE	OE.INFO_PROTECT
A.CONNECT	OE.SERIAL_LOGIN, OE.PROTECT, OE.PHYSICAL
A.COOP	OE.CREDEN
A.LOCATE	OE.PHYSICAL
A.MANAGE	OE.ADMIN, OE.INSTALL
A.NET_COMP	OE.PROTECT, OE.INSTALL
A.PEER	OE.INSTALL
A.PROTECT	OE.INFO_PROTECT, OE.PHYSICAL
A.SENSITIVITY	OE.INFO_PROTECT
A.UTRAIN	OE.INFO_PROTECT
A.UTRUST	OE.INFO_PROTECT

Table 19: Mapping Policies to Objectives

Policy	Objective
P.ACCOUNTABILITY	O.AUDITING, O.MANAGE, O.ENFORCEMENT
P.AUTHORIZED_USERS	O.AUTHORIZATION, O.MANAGE, O.ENFORCEMENT, O.STACK
P.DATAFLOW	O.MANDATORY_ACCESS, O.MANAGE, O.ENFORCEMENT, O.NETWORK_ACCESS, O.RESIDUAL_INFORMATION
P.DIST_USERS	O.ENFORCEMENT
P.ERASE	O.ERASE
P.INTEGRITY	O.MANDATORY_INTEGRITY
P.NEED_TO_KNOW	O.DISCRETIONARY_ACCESS, O.MANAGE, O.ENFORCEMENT, O.RESIDUAL_INFORMATION, OE.SOFTWARE_IN
P.STATIC	OE.INSTALL
P.TCBINTEGRITY	O.TCB_ACCESS

8.2.2 Security Objectives Sufficiency

T.UAACESS: 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_INFORMATION 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. O.TCB_ACCESS contributes further by providing a trusted computing base that allows modifications to TCB files only in maintenance (single user) mode. O.STACK ensures that a user cannot hijack a trusted process through the use of a buffer overflow type attack to gain access to a resource without the resource owner's permission.

T.UAACTION: The threat of undetected security policy violation is removed by O.AUDITING requiring the TOE to collect evidence of security relevant actions and make it accessible to authorized administrators. O.MANAGE provides a system administrator with the capability to manage the audit system such that it is capable to monitor all critical aspects of the security policy.

T.UAUSER: The threat of impersonation of an authorized user by an attacker is sufficiently diminished by O.AUTHORIZATION requiring proper authorization of users gaining access to the TOE and O.AUDITING which requires the collection of evidence of security relevant actions, which includes authorization attempts. O.MANAGE ensures that only authorized administrators (which are assumed to be trustworthy) have the ability to add new users or modify the attributes of users. O.STACK ensures that a user (authorized or unauthorized) of a trusted process cannot maliciously execute code placed on the process stack through a buffer overflow type attack. Together those objectives ensure that no unauthorized user can impersonate as an authorized user.

T.VIOS: The threat of a VIOS SCSI device driver acting on behalf of an LPAR partition attempting to access a logical volume that's not assigned to the partition is removed by O.VIOS which provides access control between VIOS SCSI device drivers and logical volumes. Similarly, the threat of a VIOS Ethernet device driver acting on behalf of a group of LPAR partitions attempting to access a VIOS Ethernet adapter device driver and vise versa is removed by O.VIOS which provides access control between the two entries.

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 a networked environment and OE.INFO_PROTECT requiring procedures for the appropriate protection of 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.

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.LPAR: The threat that software running in another logical partition than the TOE itself, therefore having different hardware resource of the same machine assigned to it than the TOE, is addressed by the objective OE.LPAR, requiring from the IT environment (i.e. the underlying machine) to successfully restrict access to resources that are assigned to one logical partition to the operating system running in that partition, therefore preventing access from software in other partitions to the TOE's logical partition.

A. NO_EVIL_ADM: 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.CLEARANCE: The assumption on procedures for managing security clearances of users is met by OE.INFO_PROTECT requiring to establish such procedures.

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

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.

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 distributed 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 distributed system.

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.SENSITIVITY: The assumption on procedures for establishing and marking information with its security levels during import and export is met by OE.INFO_PROTECT requiring to establish such procedures.

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.ACCOUNTABILITY (see LSPP section 7.1.2): The policy demanding accounting for user actions is implemented by O.AUDITING requiring auditing for security relevant actions and supported by O.MANAGE for the management of this functionality and O.ENFORCEMENT ensuring the correct invocation.

P.AUTHORIZED_USERS (see LSPP section 7.1.2): 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 these functions and O.ENFORCEMENT and O.STACK ensuring the correct invocation of the functions.

P.DATAFLOW (see LSPP section 7.1.2): The policy demanding access control based on sensitivity label is implemented by O.MANDATORY_ACCESS requiring labeling and associated access control. The O.RESIDUAL_INFORMATION objective ensures that information will not given to users that do not have a cleared access, when resources are reused. The O.MANAGE supports this policy by requiring only authorized administrators manage the functions and O.ENFORCEMENT ensures that functions are invoked and operate correctly. The O.NETWORK_ACCESS security objective supports this policy when the TOE communications with other systems across a network.

P.ERASE: The policy asks for the provision of a hard disk erase function, which is implemented by O.ERASE requiring the TOE to offer overwriting of hard disk drives with bit patterns that prevent the recovery of the original information stored on the disks.

P.DIST_USERS: The policy ensures that user accounts are defined consistently across all instances of the TOE in a distributed environment. This allows all the TOEs to consistently enforce access control across all TOE instances as supported by O.ENFORCEMENT especially in the case where NFS is involved.

P.NEED_TO_KNOW (see LSPP section 7.1.2): 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. It is supported by O.RESIDUAL_INFORMATION ensuring that resources do not release such information during reuse and by OE.SOFTWARE_IN preventing users other than authorized administrators from installing new software that might affect the access control functionality. O.MANAGE allows administrators to manage this functions, O. ENFORCEMENT ensures that the functions are invoked and operate correctly.

P.STATIC: The policy ensures, by demanding appropriate organizational measures, that no additional object reuse issues are imposed with the support of logical partitioning by the TOE: while the TOE provides functionality to support the dynamic allocation and release of hardware resources during operation, such dynamic partitioning by the use of a Hardware Management Console for the underlying hardware must not be performed by an administrator while the TOE

is running. This is implemented by OE.INSTALL, demanding a secure installation and configuration of TOE systems, which applies also to the underlying hardware.

P.TCBINTEGRITY: The integrity of the information belonging to the Trusted Computing Base is protected and verified by the objective O.TCB_ACCESS, which mandates the protections of files belonging to the TCB.

8.3 Security Requirements Rationale

SFRs for the TOE have been selected in addition to LSPP in this ST to address additional functionality provided by the TOE without conflicting with LSPP's TSP.

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 AIX. Therefore no specific security functional requirements and security functions for the non-IT environment have been defined in this Security Target. Individual sites running AIX 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. Section 8.3.3 provides more detailed rationale for the selection of the security functional requirements for the IT environment.

8.3.1 Security Requirements Coverage

Section 7.2.2 of the Labeled Security Protection Profile provides a table mapping security objectives for the TOE to the security functional requirements. In addition this section of the Protection Profile also provides a discussion with the detailed evidence of the coverage for each security objective. In accordance with [GUIDE] this rationale is seen as sufficient for discussion of the coverage of security requirements for those objectives and security requirements taken directly from the Protection Profile.

In addition to the rationale provided in the Protection Profile the following table shows that each security functional requirement, including the SFRs that have been selected in addition to those from LSPP, addresses at least one objective.

Table 20: Mapping Security Functional Requirements to Objectives

SFR	Objectives
FAU_GEN.1	O.AUDITING
FAU_GEN.2	O.AUDITING
FAU_SAR.1	O.AUDITING O.MANAGE
FAU_SAR.2	O.AUDITING
FAU_SAR.3	O.AUDITING O.MANAGE
FAU_SEL.1	O.AUDITING O.MANAGE
FAU_STG.1	O.AUDITING
FAU_STG.3	O.AUDITING O.MANAGE
FAU_STG.4	O.AUDITING O.MANAGE
FDP_ACC.1(1)	O.DISCRETIONARY_ACCESS
FDP_ACC.1(2)	O.TCB_ACCESS

SFR	Objectives
FDP_ACC.1(3)	O.AUTHORIZATION
FDP_ACC.1(4)	O.VIOS
FDP_ACF.1(1)	O.DISCRETIONARY_ACCESS
FDP_ACF.1(2)	O.TCB_ACCESS
FDP_ACF.1(3)	O.AUTHORIZATION
FDP_ACF.1(4)	O.VIOS
FDP_ETC.1	O.MANDATORY_ACCESS
FDP_ETC.2	O.MANDATORY_ACCESS
FDP_IFC.1(1)	O.MANDATORY_ACCESS
FDP_IFC.1(2)	O.MANDATORY_INTEGRITY
FDP_IFC.1(3)	O.NETWORK_ACCESS
FDP_IFF.2(1)	O.MANDATORY_ACCESS
FDP_IFF.2(2)	O.NETWORK_ACCESS
FDP_IFF.2(3)	O.MANDATORY_INTEGRITY
FDP_ITC.1	O.MANDATORY_ACCESS
FDP_ITC.2	O.MANDATORY_ACCESS
FDP_RIP.2	O.RESIDUAL_INFORMATION
Note 1	O.RESIDUAL_INFORMATION
FDP_RIP.3-AIX	O.ERASE
FIA_ATD.1(1)	O.AUTHORIZATION O.DISCRETIONARY_ACCESS O.MANDATORY_ACCESS O.MANDATORY_INTEGRITY O.AUDITING O.TCB_ACCESS O.NETWORK_ACCESS
FIA_ATD.1(2)	O.AUTHORIZATION
FIA_SOS.1	O.AUTHORIZATION
FIA_UAU.2	O.AUTHORIZATION
FIA_UAU.7	O.AUTHORIZATION
FIA_UID.2	O.AUTHORIZATION
FIA_USB.1(1)	O.DISCRETIONARY_ACCESS O.AUDITING O.AUTHORIZATION O.MANDATORY_ACCESS O.MANDATORY_INTEGRITY O.TCB_ACCESS O.NETWORK_ACCESS
FIA_USB.1(2)	O.AUTHORIZATION
FMT_MSA.1(1)	O.DISCRETIONARY_ACCESS O.MANDATORY_ACCESS
FMT_MSA.1(2)	O.NETWORK_ACCESS
FMT_MSA.1(3)	O.TCB_ACCESS
FMT_MSA.1(4)	O.MANDATORY_INTEGRITY
FMT_MSA.1(5)	O.AUTHORIZATION
FMT_MSA.1(6)	O.VIOS
FMT_MSA.3(1)	O.DISCRETIONARY_ACCESS O.MANDATORY_ACCESS

SFR	Objectives
FMT_MSA.3(2)	O.NETWORK_ACCESS
FMT_MSA.3(3)	O.TCB_ACCESS
FMT_MSA.3(4)	O.MANDATORY_INTEGRITY
FMT_MSA.3(5)	O.AUTHORIZATION
FMT_MSA.3(6)	O.VIOS
FMT_MTD.1(1)	O.AUDITING O.MANAGE
FMT_MTD.1(2)	O.AUDITING O.MANAGE
FMT_MTD.1(3)	O.AUDITING O.MANAGE
FMT_MTD.1(4)	O.MANAGE
FMT_MTD.1(5)	O.AUTHORIZATION O.MANAGE
FMT_MTD.1(6)	O.ENFORCEMENT O.MANAGE
FMT_MTD.1(7)	O.AUTHORIZATION O.MANAGE
FMT_REV.1(1)	O.MANAGE
FMT_REV.1(2)	O.DISCRETIONARY_ACCESS O.MANDATORY_ACCESS
FMT_REV.1(3)	O.MANAGE
FMT_SMF.1	O.MANAGE
FMT_SMR.1(1)	O.MANAGE
FMT_SMR.1(2)	O.MANAGE, O.VIOS
FPT_AMT.1	O.ENFORCEMENT
FPT_RVM.1	O.ENFORCEMENT
FPT_RVM.2-AIX	O.STACK
FPT_SEP.1	O.ENFORCEMENT
FPT_STM.1	O.AUDITING
FPT_TDC.1	O.NETWORK_ACCESS
FPT_TST.1	O.ENFORCEMENT O.TCB_ACCESS

Table 21: 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
FDP_ACC.1(LPAR)	OE.LPAR
FDP_ACF.1(LPAR)	OE.LPAR
FIA_UID.2	OE.LPAR

8.3.2 Security Requirements Sufficiency for the TOE

This section discusses how the single objectives defined for the TOE are met by the SFRs selected in this ST. Whenever such rationale has already been sufficiently shown in LSPP for an objective, only the additional aspects introduced by augmentations to LSPP in this ST are discussed.

O.AUDITING is implemented by the SFRs discussed in LSPP. In addition, it is supported by the security attributes required in FIA_ATD.1(1) and the management actions for auditing specified in FMT_MTD.1(3).

O.AUTHORIZATION is implemented by the SFRs discussed in LSPP. An additional authorization mechanism enforced by the TOE is modeled as a discretionary access control policy in FDP_ACF.1(3) and called out in FDP_ACC.1(3), with supporting management functions in FMT_MSA.1(5) and FMT_MSA.3(5) and the security attributes required in FIA_ATD.1(1). For VIOS, the security attributes are defined by FIA_ATD.1(2) and the user-subject binding by FIA_USB.1(2).

O.DISCRETIONARY_ACCESS is implemented by the SFRs discussed in LSPP.

O.ENFORCEMENT is implemented by the SFRs discussed in LSPP and supported by the additional privilege enforcement mechanism implemented in the TOE, which are represented by the requirements in FPT_RVM.1 for a reference mechanism and additional management functionality in FMT_MTD.1(6). Verification of the TCB access control (see O.TCB_ACCESS) is supported by functionality to test the TCB's integrity in FPT_TST.1.

O.ERASE is addressed by providing a mechanism to overwrite residual information on hard disk drives upon request of administrators in FDP_RIP.3-AIX.

O.MANAGE is implemented by the SFRs discussed in LSPP and by additional management functionality for the security functions that have been introduced in addition to the LSPP requirements in FMT_MTD.1(3) and FMT_MTD.1(6). Security function management is called out in FMT_SMF.1. For VIOS, the roles are defined by FMT_SMR.1(2) with user attribute revocation defined by FMT_REV.1(3).

O.MANDATORY_ACCESS is implemented by the SFRs discussed in LSPP, supported by security attributes required in FIA_ATD.1(1) and rules for revocation of security attributes related to MAC in FMT_REV.1(2).

O.MANDATORY_INTEGRITY is implemented by SFRs defining the mandatory integrity control policy in FDP_IFC.1(2) and FDP_IFF.2(3), user-subject binding in FIA_USB.1(1) and security attributes spelled out in FIA_ATD.1(1), as well as management functionality in FMT_MSA.1(4) and FMT_MSA.3(4).

O.NETWORK_ACCESS is implemented by the TOE's advanced secure networking policy, which has been defined in FDP_IFC.1(3) and FDP_IFF.2(2), with security attributes in FIA_ATD.1(1) and user-subject binding in FIA_USB.1(1) and management functionality in FMT_MSA.1(2) and FMT_MSA.3(2). A requirement for consistent interpretation of security labels in networked sessions is spelled out in FPT_TDC.1.

O.RESIDUAL_INFORMATION is implemented by the SFRs discussed in LSPP.

O.STACK is implemented by the TOE's Stack Execution Disable (SED) feature which allows an authorized administrator to disable the execution of code residing on the stack of selected processes in FPT_RVM.2-AIX.

O.TCB_ACCESS is implemented by the TOE's trusted computing base, which is modeled in FDP_ACC.1(2) and FDP_ACF.1(2), with security attributes in FIA_ATD.1(1) and user-subject binding in FIA_USB.1(1) and management functionality in FMT_MSA.1(3) and FMT_MSA.3(3) and a mechanism for testing the TCB integrity in FPT_TST.1.

O.VIOS is implemented by the TOE and modeled in FDP_ACC.1(4) and FDP_ACF.1(4) with management functionality in FMT_MSA.1(6) and FMT_MSA.3(6). The roles are defined by FMT_SMR.1(2).

8.3.3 Rationale for Security Requirements for the IT environment

In addition to the requirements of [LSPP] this Security Target has added 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 a PowerPC based processor those rules have to be defined precisely for the specific processor type that is the target of the hardware evaluation.

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

In addition, the security requirements FDP_ACC.1 (LPAR) and FDP_ACF.1 (LPAR) have been chosen to express the need for an access control policy implemented in the underlying hardware to regulate access to parts of the hardware that are assigned to different logical partitions (LPARs). If an LPAR enabled underlying machine allows to run several operating systems in different logical partitions, with dedicated hardware resources assigned to those partitions, means are required to prevent the operating system running in one partition from accessing the resources assigned to another operating system running on the same machine. The hypervisor must also be able to identify each LPAR partition in order to allow communication channels to be setup between partitions (FIA_UID.2) as well as to know the resources allocated to each partition.

Since the underlying hardware for the TOE provides LPAR support, access to the TSF and TSF data from other logical partitions than the one that belongs to the TOE must be prevented. Such protection has to be provided by the IT environment, which is expressed in the security requirements meeting the objective OE.LPAR.

8.3.4 Justification of explicitly expressed security requirements

The explicit requirement Note 1 is adopted from the LSPP, the substantiation of the LSPP applies: The CC's FDP_RIP components only specify resources being allocated to objects and does not address resources used directly by subjects, such as memory or registers. This explicit requirement was added to ensure coverage of these resources. The words are identical to FDP_RIP.2 except "subject" replaces "object".

The name "Note 1" has been adopted from the Protection Profile for an easy mapping to the requirements defined there. It is known to the authors of this Security Target that this name is not compliant with the recommendations for the naming of components additional to the ones defined in part two of the CC.

The explicit requirement FDP_RIP.3-AIX has been introduced as a response to the objective O.ERASE. While it might have been possible to rush the implementation of this objective by using the existing components from the FDP_RIP family, the ST author felt that it was necessary to distinguish between the object reuse properties that are inherent in the management of shared resources of a TOE, as defined in FDP_RIP.1 and .2, and the administrator-invoke-able functionality to make residual information unavailable "on demand" before e.g. removing a resource from a system.

The explicit requirement FPT_RVM.2-AIX has been introduced in response to the objective O.STACK. This SFR describes an extended capability of AIX that improves the reference mediation of AIX beyond that described by FPT_RVM.1. Specifically, it helps prevent the misuse of the TOE (specifically trusted processes) against its own programmatic shortcomings and possible compromise.

8.3.5 Security Requirements Dependency Analysis

This dependency analysis takes into account only the SFRs that have been selected in addition to those derived from LSPP. With respect to dependencies in LSPP, the ST authors rely on the evaluation of the registered PP, noting that since then a dependency of components in LSPP on FMT_SMF.1 has been introduced and is fulfilled by this ST, and that FIA_UAU.1 and FIA_UID.1 as defined in the LSPP have been replaced by the hierarchically higher components FIA_UAU.2 and FIA_UID.2. FIA_UAU.1 and FIA_UAU.2 have the same dependency on FIA_UID.1, which is resolved by the inclusion of FIA_UID.2, which is hierarchical to FIA_UID.1. FIA_UID.1 and FIA_UID.2 both have no dependency.

Since no other additional security functional requirements to those defined in the LSPP have been defined for the TOE, the dependency analysis for the security functional requirements in section 7.3 of the Protection Profile applies for all security requirements taken from the LSPP.

Table 22: Dependency analysis for the TOE SFRs

SFR	Origin	Dependencies	Resolved?
FDP_ACC.1(2)	CC Part 2	FDP_ACF.1	FDP_ACF.1(2)
FDP_ACC.1(3)	CC Part 2	FDP_ACF.1	FDP_ACF.1(3)
FDP_ACC.1(4)	CC Part 2	FDP_ACF.1	FDP_ACF.1(4)
FDP_ACF.1(2)	CC Part 2	FDP_ACC.1 FMT_MSA.3	FDP_ACC.1(2) FMT_MSA.3(3)
FDP_ACF.1(3)	CC Part 2	FDP_ACC.1 FMT_MSA.3	FDP_ACC.1(3) FMT_MSA.3(5)
FDP_ACF.1(4)	CC Part 2	FDP_ACC.1 FMT_MSA.3	FDP_ACC.1(4) FMT_MSA.3(6)
FDP_IFC.1(2)	CC Part 2	FDP_IFF.1	FDP_IFF.2(3)
FDP_IFC.1(3)	CC Part 2	FDP_IFF.1	FDP_IFF.2(2)
FDP_IFF.2(2)	CC Part 2	FDP_IFC.1 FMT_MSA.3	FDP_IFC.1(3) FMT_MSA.3(2)
FDP_IFF.2(3)	CC Part 2	FDP_IFC.1 FMT_MSA.3	FDP_IFC.1(2) FMT_MSA.3(4)
FDP_RIP.3-AIX	ECD (section 5.1)	None	yes
FIA_ATD.1(2)	CC Part 2	None	yes
FIA_USB.1(2)	CC Part 2	FIA_ATD.1	FIA_ATD.1(2)
FMT_MSA.1(2)	CC Part 2	[FDP_ACC.1 or FDP_IFC.1] FMT_SMF.1 FMT_SMR.1	FDP_IFC.1(3) FMT_SMF.1 FMT_SMR.1(1)
FMT_MSA.1(3)	CC Part 2	[FDP_ACC.1 or FDP_IFC.1] FMT_SMF.1 FMT_SMR.1	FDP_ACF.1(2) FMT_SMF.1 FMT_SMR.1(1)
FMT_MSA.1(4)	CC Part 2	[FDP_ACC.1 or FDP_IFC.1] FMT_SMF.1 FMT_SMR.1	FDP_IFC.1(1) FMT_SMF.1 FMT_SMR.1(1)
FMT_MSA.1(5)	CC Part 2	[FDP_ACC.1 or FDP_IFC.1] FMT_SMF.1 FMT_SMR.1	FDP_ACF.1(3) FMT_SMF.1 FMT_SMR.1(1)
FMT_MSA.1(6)	CC Part 2	[FDP_ACC.1 or FDP_IFC.1] FMT_SMF.1 FMT_SMR.1	FDP_ACF.1(4) FMT_SMF.1 FMT_SMR.1(2)
FMT_MSA.3(2)	CC Part 2	FMT_MSA.1 FMT_SMR.1	FMT_MSA.1(2) FMT_SMR.1(1)
FMT_MSA.3(3)	CC Part 2	FMT_MSA.1 FMT_SMR.1	FMT_MSA.1(3) FMT_SMR.1(1)
FMT_MSA.3(4)	CC Part 2	FMT_MSA.1	FMT_MSA.1(4)

SFR	Origin	Dependencies	Resolved?
		FMT_SMR.1	FMT_SMR.1(1)
FMT_MSA.3(5)	CC Part 2	FMT_MSA.1 FMT_SMR.1	FMT_MSA.1(5) FMT_SMR.1(1)
FMT_MSA.3(6)	CC Part 2	FMT_MSA.1 FMT_SMR.1	FMT_MSA.1(6) FMT_SMR.1(2)
FMT_MTD.1(3)	CC Part 2	FMT_SMF.1 FMT_SMR.1	FMT_SMF.1 FMT_SMR.1(1)
FMT_MTD.1(6)	CC Part 2	FMT_SMF.1 FMT_SMR.1	FMT_SMF.1 FMT_SMR.1(1)
FMT_MTD.1(7)	CC Part 2	FMT_SMF.1 FMT_SMR.1	FMT_SMF.1 FMT_SMR.1(2)
FMT_REV.1(3)	CC part 2	FMT_SMR.1	FMT_SMR.1(2)
FMT_SMF.1	CC Part 2	None	yes
FMT_SMR.1(2)	CC Part 2	FIA_UID.1	FIA_UID.2
FPT_RVM.2-AIX	CC Part 2	None	yes
FPT_TDC.1	CC Part 2	None	yes
FPT_TST.1	CC Part 2	FPT_AMT.1	FPT_AMT.1

Table 23: Dependency analysis for IT environment SFRs

SFR	Origin	Dependencies	Resolved?
FDP_ACC.1	CC Part 2	FDP_ACF.1	FDP_ACF.1
FDP_ACF.1	CC Part 2	FDP_ACC.1 FMT_MSA.3	FDP_ACC.1 FMT_MSA.3
FMT_MSA.3	CC Part 2	FMT_MSA.1 FMT_SMR.1	no
FDP_ACC.1(LPAR)	CC Part 2	FDP_ACF.1	FDP_ACF.1(LPAR)
FDP_ACF.1(LPAR)	CC Part 2	FDP_ACC.1 FMT_MSA.3	FDP_ACC.1(LPAR) no
FIA_UID.2	CC Part 2	None	yes

The dependency analysis for the pre-defined EAL4 has already been performed by the creators of the Common Criteria and is not repeated in this section. ALC_FLR.1, which has been added as a security assurance requirement has no dependency on any security functional or security assurance requirement. Therefore the dependency analysis for EAL4 of the Common Criteria applies.

Since EAL4 is hierarchical to EAL3, all dependencies of security functional requirements on assurance requirements listed in the Protection Profile are also resolved.

8.3.6 Justification of unresolved dependencies

As demonstrated in the dependency analysis above, all dependencies between security requirements for the TOE are resolved since all the dependencies within the LSPP are resolved, dependencies of additional SFRs have been shown to be completely taken into account, EAL 4 as a pre-defined evaluation assurance level contains no unresolved references to other assurance components and no additional references to functional components and ALC_FLR.1 has no dependencies.

The dependencies of FMT_MSA.3 on FMT_MSA.1 and FMT_SMR.1 for the security requirements for the IT environment are not resolved, because the processor does not allow to “manage” the use of the processor attribute and there is no role model involved. The processor switches between “user” and “supervisor” mode under well defined conditions where the TSF defined in this Security Target are required to “manage” those conditions. Roles (especially human roles) are not involved here.

The dependency on FMT_MSA.3 from FDP_ACF.1(LPAR) for the IT environment has not been resolved in this ST, since imposing a decision whether the management of LPAR resources during TOE operation should be manageable would be an unnecessary restriction for the ST author of the underlying system.

8.3.7 Strength of function

This Security Target claims in compliance with LSPP a SOF rating SOF-medium. This claim applies for FIA_SOS.1, whereby LSPP states 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.

8.3.8 Evaluation Assurance Level

The EAL defined in the LSPP is EAL3, as LSPP addresses a generalized environment with a moderate level of risk to the assets. This security target claims EAL4 augmented with ALC_FLR.1, meeting this assumptions on the environment as well by providing a higher evaluation assurance level.

8.4 TOE Summary Specification Rationale

8.4.1 Security Functions Justification

The following table demonstrates the relationship between the TSF and the individual TSF aspects described in the TSS and security functional requirements for the TOE from chapter 5.

Table 24: Mapping of TSF and TSF aspects to SFRs

Security Function or Security Function Aspects	Corresponding SFRs
IA – Identification and Authentication	
IA.1	FIA_ATD.1(1), FIA_ATD.1(2), FIA_SOS.1, FMT_MTD.1(4), FMT_SMF.1
IA.2	FAU_GEN.2, FIA_UAU.2, FIA_UID.2
IA.3	FAU_GEN.2, FIA_UAU.2, FIA_UID.2, FIA_UAU.7
IA.4	FAU_GEN.2, FIA_USB.1(1), FIA_USB.1(2)
IA.5	FIA_USB.1(1), FIA_USB.1(2)
IA.6	FIA_USB.1(1), FIA_USB.1(2)
AU – Auditing	
AU.1	FAU_GEN.1, FAU_GEN.2
AU.2	FAU_GEN.1, FAU_GEN.2
AU.3	FAU_SAR.1, FAU_SAR.3
AU.4	FAU_SAR.1, FAU_SAR.3
AU.5	FAU_SAR.2, FAU_STG.1, FMT_MTD.1(1)
AU.6	FAU_STG.3, FAU_STG.4
AU.7	FAU_SAR.2
DA – Discretionary Access Control	
DA.1	FDP_ACC.1(1), FDP_ACF.1(1)
DA.2	FDP_ACC.1(1), FDP_ACF.1(1)
DA.3	FDP_ACC.1(1), FDP_ACF.1(1), FMT_MSA.1(1), FMT_MSA.3(1), FMT_SMF.1
DA.4	FDP_ACC.1(1), FDP_ACF.1(1), FMT_MSA.1(1), FMT_MSA.3(1), FMT_SMF.1
DA.5	FDP_ACC.1(4), FDP_ACF.1(4), FMT_MSA.1(6), FMT_MSA.3(6), FMT_MTD.1(7), FMT_SMF.1
PV – Privileges	
PV.1	FAU_SAR.2, FDP_ACC.1(1), FDP_ACC.1(2), FDP_ACC.1(3), FDP_ACF.1(1), FDP_ACF.1(2), FDP_ACF.1(3), FDP_IFC.1(1), FDP_IFC.1(2), FDP_IFC.1(3), FDP_IFF.1(1), FDP_IFF.1(2), FDP_IFF.1(3), FMT_MTD.1(6)
PV.2	FAU_SAR.2, FDP_ACC.1(1), FDP_ACC.1(2), FDP_ACC.1(3), FDP_ACF.1(1), FDP_ACF.1(2), FDP_ACF.1(3), FDP_IFC.1(1), FDP_IFC.1(2), FDP_IFC.1(3), FDP_IFF.1(1), FDP_IFF.1(2),

Security Function or Security Function Aspects	Corresponding SFRs
	FDP_IFF.1(3)
PV.3	FAU_SAR.2, FDP_ACC.1(1), FDP_ACC.1(2), FDP_ACC.1(3), FDP_ACF.1(1), FDP_ACF.1(2), FDP_ACF.1(3), FDP_IFC.1(1), FDP_IFC.1(2), FDP_IFC.1(3), FDP_IFF.1(1), FDP_IFF.1(2), FDP_IFF.1(3)
AZ – Authorizations	
AZ.1 – AZ.5	FDP_ACC.1(3), FDP_ACF.1(3)
MAC – Mandatory Access Control	FDP_ETC.1, FDP_ETC.2, FDP_IFC.1(1), FDP_IFF.2(2), FDP_ITC.1, FDP_ITC.2
ASN – Advanced Secure Networking	
ASN.1	FDP_ETC.1, FDP_ETC.2, FDP_IFC.1(3), FDP_IFF.2(3), FDP_ITC.1, FDP_ITC.2
ASN.2	FDP_ETC.2, FDP_ITC.1, FPT_TDC.1
MIC – Mandatory Integrity Checking	
MIC.1	FDP_IFC.1(2), FDP_IFF.2(2)
OR – Object Reuse	
OR.1	FDP_RIP.2, Note 1
OR.2	FDP_RIP.2, Note 1
OR.3	FDP_RIP.2, Note 1
OR.4	FDP_RIP.2, Note 1
OR.5	FDP_RIP.3-AIX
SM – Security Management	
SM.1	FMT_SMR.1(1), FMT_SMR.1(2)
SM.2	FAU_GEN.1, FAU_SEL.1, FMT_MTD.1(1), FMT_MTD.1(2), FMT_SMF.1
SM.3	FMT_MSA.1(1)-(6), FMT_MSA.3(1)-(6), FMT_MTD.1(7), FMT_SMF.1, FMT_REV.1(2)
SM.4	FIA_ATD.1(1), FIA_ATD.1(2), FIA_SOS.1, FMT_MTD.1(4), FMT_MTD.1(5), FMT_SMF.1, FMT_REV.1(1), FMT_REV.1(3)
SM.5	FPT_STM.1
TP – TSF Protection	
TP.1	FPT_RVM.1
TP.2	FPT_SEP.1
TP.3	FPT_SEP.1
TP.4	FIA_UAU.2, FIA_UAU.7, FIA_UID.2, FPT_SEP.1
TP.5	FAU_SEL.1, FMT_MSA.3(1)-(5), FMT_MTD.1(1)-(5), FMT_SMF.1
TP.6	FPT_SEP.1
TP.7	FAU_GEN,1. FPT_AMT.1
TP.8	FPT_TST.1
TP.9	FAU_SAR.2, FAU_SEL.1, FDP_ACC.1(2), FDP_ACF.1(2), FDP_IFC.1(1), FDP_IFF.2(1)

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 25: Mapping Security Functional Requirements to Security Functions

SFR	Security Functions (TOE summary specification)
FAU_GEN.1	The requirement for the information to be recorded with an audit event are satisfied by the generation of audit data is fulfilled by the security functions AU.1 specifying the audit record format and SM.2 describing the audit control files which are used to define the events that can be audited. AU.2 describes the process used within the TOE to generate an audit

SFR	Security Functions (TOE summary specification)
	record. The results of diagnostic tests are stored in a separate error log file as described in TP.7.
FAU_GEN.2	The association of auditable events with its causing identity is done in AU.1 specifying the audit record format, including the user and login ID of the creator of the auditable event, and AU.2 generating the audit record. IA.2 and IA.3 describe the authentication process which ensures that the ID of the a user is authenticated. IA.4 describes that although a user may change his / her effective user ID, the login user ID (which is recorded in the audit record) can not be changed. This ensures that the ID the user has used when he has authenticated to the system is recorded with every audit event that this user has caused.
FAU_SAR.1	The system administrator can use the commands described in AU.3 and AU.4 to select, read, process and print audit records.
FAU_SAR.2	Read access to the audit records is granted only to explicitly privileged users as enforced by the audit file protection of AU.5. In addition, privileges are used to enforce restrictions with respect to audit management as in AU.7 and in PV.1, PV.2 and PV.3. File security flags are used to identify audit-related files and restrict access to them (TP.9).
FAU_SAR.3	The requirement to allow searches of the audit data based on specified attributes is met by the audit review function described in AU.3 and AU.4. auditselect can be used as a tool to select audit records using an expression based on the audit record fields listed in table 6.2. This table contains all the selection criteria listed in FAU_SAR.3.
FAU_SEL.1	The in- or exclusion of auditable events from the set of audited events is provided by SM.2 (audit configuration and management). SM.2 describes how a system administrator can select the events to be audited based on event type, file name and user identity and defines the system configuration files used in this function System configuration files in general are also described in TP.5. A file security flag can be used to override the decision not to audit access to individual files (TP.9).
FAU_STG.1	Audit data is protected by AU.5. This prevents audit records to be deleted or modified by other users than the system administrator.
FAU_STG.3	AU.6 describes the process AIX takes when the audit trail exceeds a defined threshold.
FAU_STG.4	Prevention of audit data loss is described in AU.6. The system can be configured to go into “panic” mode and stop the host when the audit trail is full.
FDP_ACC.1(1)	The discretionary access control policy is based on DA.1 and DA.2 defining permission bits for the subjects and objects as in DA.3 for file system objects, and DA.4 for IPC objects. This is supported by additional restrictions enforced with the privilege mechanism, which is detailed in PV.1, PV.2 and PV.3.
FDP_ACC.1(2)	The trusted computing base model is implemented in TP.9. This is supported by additional restrictions enforced with the privilege mechanism, which is detailed in PV.1, PV.2 and PV.3, and by file security flags described in TP.9.
FDP_ACC.1(3)	The implementation of the authorizations mechanism is described in AZ.1 through AZ.5. This is supported by additional restrictions enforced with the privilege mechanism, which is detailed in PV.1, PV.2 and PV.3.
FDP_ACC.1(4)	The VIOS access control policy is based on DA.5 defining the mapping of LPAR partitions to logical/physical volumes and defining the mapping of Ethernet packets to groups of LPAR partitions sharing a virtual network.
FDP_ACF.1(1)	The discretionary access control is realized as described above by DA.1, DA.2, DA.3, and DA.4. There the individual mechanisms for access control depending on the object type are described in detail. This is supported by additional restrictions enforced with the privilege mechanism, which is detailed in PV.1, PV.2 and PV.3.
FDP_ACF.1(2)	The trusted computing base model is implemented in TP.9. This is supported by additional restrictions enforced with the privilege mechanism, which is detailed in PV.1, PV.2 and PV.3.
FDP_ACF.1(3)	The implementation of the authorizations mechanism is described in AZ.1 through AZ.5. This is supported by additional restrictions enforced with the privilege mechanism, which is detailed in PV.1, PV.2 and PV.3.
FDP_ACF.1(4)	The VIOS access control policy is based on DA.5 defining the mapping of LPAR partitions to logical/physical volumes and defining the mapping of Ethernet packets to groups of LPAR partitions sharing a virtual network.

SFR	Security Functions (TOE summary specification)
FDP_ETC.1	Export of unlabeled data is implemented in MAC (and ASN.1 for network objects).
FDP_ETC.2	Export of labeled data is implemented in MAC (and ASN.1, ASN.2 for network objects).
FDP_IFC.1(1)	The implementation of the mandatory access control policy is described in MAC. This is supported by additional restrictions enforced with the privilege mechanism, which is detailed in PV.1, PV.2 and PV.3, and by a file security flag described in TP.9.
FDP_IFC.1(2)	The mandatory integrity control policy is implemented as described in MIC.1. This is supported by additional restrictions enforced with the privilege mechanism, which is detailed in PV.1, PV.2 and PV.3.
FDP_IFC.1(3)	ASN.1 describes how the TOE implements the advanced secure networking policy. This is supported by additional restrictions enforced with the privilege mechanism, which is detailed in PV.1, PV.2 and PV.3.
FDP_IFF.2(1)	The implementation of the mandatory access control policy is described in MAC. This is supported by additional restrictions enforced with the privilege mechanism, which is detailed in PV.1, PV.2 and PV.3, and by a file security flag detailed in TP.9.
FDP_IFF.2(2)	ASN.1 describes how the TOE implements the advanced secure networking policy. This is supported by additional restrictions enforced with the privilege mechanism, which is detailed in PV.1, PV.2 and PV.3.
FDP_IFF.2(3)	The mandatory integrity control policy is implemented as described in MIC.1. This is supported by additional restrictions enforced with the privilege mechanism, which is detailed in PV.1, PV.2 and PV.3.
FDP_ITC.1	Import of unlabeled data is implemented in MAC (and ASN.1 for network objects).
FDP_ITC.2	Import of labeled data is implemented in MAC (and ASN.1, ASN.2 for network objects).
FDP_RIP.2	Object residual information protection is realized by security functions for object reuse on file system objects (OR.1), IPC objects (OR.2), queuing system objects (OR.3) and miscellaneous objects (OR.4).
Note 1	The residual information protection as realized by OR.1, OR.2, OR.3 and OR.4 (see above) applies as well to subjects.
FDP_RIP.3-AIX	Residual information protection for hard disk drives is implemented as described in OR.5.
FIA_ATD.1(1)	AIX security attributes belonging to individual users are realized by the user I&A data management of IA.1. Management of user attributes is described in SM.4.
FIA_ATD.1(2)	VIOS security attributes belonging to individual users are realized by the user I&A data management of IA.1. Management of user attributes is described in SM.4.
FIA_SOS.1	The passwd function of IA.1 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.4.
FIA_UAU.2	Authentication of each user before any action is realized by IA.2 (common authentication mechanism) and IA.3 (interactive login and related mechanisms). Authentication is initiated by a trusted process. Trusted processes are described in TP.4.
FIA_UAU.7	The login mechanisms of IA.3 provide only obscured feedback during authentication. Authentication feedback is managed by a trusted process. Trusted processes are described in TP.4.
FIA_UID.2	Identification of each user before any action is realized together with authentication as in IA.2 and IA.3 (see above). Identification is initiated by a trusted process. Trusted processes are described in TP.4.
FIA_USB.1(1)	For AIX, he required binding between subjects and users is implemented by the su functionality of IA.4 and login processing of IA.5. Function IA.6 describes the logoff process which releases the binding between subjects and users.
FIA_USB.1(2)	For VIOS, the required binding between subjects and users is implemented by the su functionality of IA.4 and login processing of IA.5. Function IA.6 describes the logoff process which releases the binding between subjects and users.
FMT_MSA.1(1)	The management of object security attributes is implemented by the access control configuration and management function SM.3, the objects are described in DA.3 (file system objects) and DA.4 (IPC objects). In addition, security attributes of relevance for MAC are addressed here.

SFR	Security Functions (TOE summary specification)
FMT_MSA.1(2)	The management of security attributes for ASN.1, ASN.2 and MAC is implemented in SM.3.
FMT_MSA.1(3)	The management of security attributes for TP.8 and TP.9 is implemented in SM.3.
FMT_MSA.1(4)	The management of security attributes for MIC is implemented in SM.3.
FMT_MSA.1(5)	The management of security attributes for AZ is implemented in SM.3.
FMT_MSA.1(6)	The management of VIOS access control policy is implemented by SM.3. The objects are described in DA.5.
FMT_MSA.3(1)	Restrictive default values for security attributes are defined for the objects when they are created. Default values can be defined by the system administrator for all object types and by the user for file system objects created under his control. (see above, e.g.,SM.3, DA.3, DA.4). Some default values are defined in TSF databases as defined in TP.5.
FMT_MSA.3(2)	The management of default values for ASN.1, ASN.2 and MAC security attributes is implemented in SM.3, while some default values are defined in TSF databases as in TP.5.
FMT_MSA.3(3)	The management of default values for TP.8 and TP.9 security attributes is implemented in SM.3, while some default values are defined in TSF databases as in TP.5.
FMT_MSA.3(4)	The management of default values for MIC security attributes is implemented in SM.3, while some default values are defined in TSF databases as in TP.5.
FMT_MSA.3(5)	The management of default values for AZ security attributes is implemented in SM.3, while some default values are defined in TSF databases as in TP.5.
FMT_MSA.3(6)	A VIOS SCSI device driver acting on behalf of an LPAR partition cannot access a logical volume or physical volume until the mapping is created in VIOS as mentioned in SM.3 with the access control described in DA.5. Similarly, a VIOS Ethernet adapter device driver cannot access a VIOS Ethernet device driver acting on behalf of a group of LPAR partitions sharing a virtual network and vise versa until the mapping is created in VIOS as mentioned in SM.3 with the access control described in DA.5.
FMT_MTD.1(1)Audit Trail	The audit trail (and the restricted access to it) is realized by the audit file protection AU.5 and the audit configuration and management SM.2. TSF databases as defined in TP.5 contain configuration parameter of the audit trail.
FMT_MTD.1(2) Audit Events	Only authorized administrators are allowed to modify or observe the set of audited events, which is implemented by the audit configuration and management SM.2. Audit attributes are stored in TSF databases described in TP.5.
FMT_MTD.1(3)	The audit threshold is managed in SM.2, using TSF databases to store the configuration as identified in TP.5.
FMT_MTD.1(4) User Attributes	User security attributes are protected as required by the user identification and authentication data management IA.1 and during the creation of new users in SM.4. User attributes are stored in TSF databases described in TP.5.
FMT_MTD.1(5) Authen. Data	Initialization of authentication data is restricted to administrators during the creation of new users in SM.4. Authentication data (in encrypted form) and attributes are stored in TSF databases described in TP.5. Users are allowed to change their own authentication data within the limits defined by the system administrator. This is described in SM.4
FMT_MTD.1(6)	Privileges are managed by authorized administrators as described as described in PV.1.
FMT_MTD.1(7)	VIOS contains the mappings of logical volumes and physical volumes to LPAR partitions. It also contains the mappings of Ethernet packets to groups of LPAR partitions sharing a virtual network. The mappings are performed as described by DA.5. Authorized administrators are allowed to modify these mappings as described in SM.3.
FMT_REV.1(1) User Attributes	The revocation of user security attributes as required in FMT_REV.1(1) is realized by the user management functions of SM.4.
FMT_REV.1(2) Obj. Attributes	Revocation of object security attributes is realized by the access control configuration and management function SM.3.
FMT_REV.1(3) VIOS User Attributes	The revocation of user security attributes as required in FMT_REV.1(3) is realized by the user management functions of SM.4.
FMT_SMF.1	Management of security functions is addressed in the following security functions: Object security attributes management: DA.3 (File system objects), DA.4 (IPC objects). In addition SM.3 defines some management functions, and the implementations of the respective access control policies in AZ, MAC, TP.9, MIC.1+.2, ASN.1, and PV.1-.3.

SFR	Security Functions (TOE summary specification)
	User attribute management: SM.4 Authentication management: SM.4 and IA.1 Audit trail management: SM.2 Audit event management: SM.2 In addition most of the management functions use the TSF databases (TP.5) to store management configurations.
FMT_SMR.1(1)	The required roles are maintained within the security management of the roles in function SM.1.
FMT_SMR.1(2)	The required roles are maintained within the security management of the roles in function SM.1.
FPT_AMT.1	Diagnostic functions are provided by TP.7 to allow abstract machine testing.
FPT_RVM.1	The TSF invocation guarantee functionality TP.1 ensure that TSP enforcement functions are always invoked before functions in the TSC are allowed to proceed.
FPT_RVM.2-AIX	The ability to prevent the execution of code on the stack of a process is enforced by the memory management mechanism of the kernel mentioned in TP.1.
FPT_SEP.1	The required domain separation for the TSF is realized by the kernel functionality TP.2 itself, the kernel extensions TP.3, trusted processes TP.4, the discretionary access control mechanism DA.3 and internal TOE protection mechanisms TP.6.
FPT_STM.1	Reliable time stamps are provided by the protected system clock of the time management function SM.5.
FPT_TDC.1	Labeling for network connections implemented in ASN.2 provides for TSF data consistency.
FPT_TST.1	TP.8 provides a mechanism to check the integrity of TSF-relevant files by comparing various critical attributes with a reference database (which is also identified in TP.5). Similarly, TP.8 provides a command to check the status of the TCB.

The above table shows how the security functions satisfy the security functional requirements for the TOE. The following rationale demonstrates how the SFRs support each other and how the security functions work together to support the SFRs.

As outlined in the TSS, the TOE implements a variety of access control mechanisms that provide for both mandatory and discretionary means to restrict access to information assets to authorized users. These access control mechanisms work together to implement a number of objectives:

- The Discretionary Access Control is specified in FDP_ACC.1(1) and FDP_ACF.1(1) and implemented as described for the security function DA.
- The Mandatory Access Control policy is specified in FDP_IFC.1(1) and FDP_IFF.2(1) and implemented in MAC. Special rules apply for the im- and export of labeled (FDP_ITC.2, FDP_ETC.2) and unlabeled (FTP_ITC.1, FDP_ETC.1) data.
- A separate policy has been specified in FDP_IFC.1(3) and FDP_IFF.2(2) for mandatory access control in network transactions, which is implemented as part of the security function MAC and provides for the consistency of labels between the TOE and remote systems (FPT_TDC.1).
- A Mandatory Integrity Control policy is modeled in FDP_IFC.1(2) and FDP_IFF.2(3) and implemented as in MIC.
- As additional integrity protection for elementary system resources, the TOE implements a trusted computing base as described in TP.8 and TP.9, which is specified in FDP_ACC.1(2) and FDP_ACF.1(2) and restricts modifications to certain TSF-relevant files to the maintenance mode of the system. Integrity checks for the TCB files are offered as modeled in FPT_TST.1.

The TOE offers a mechanism to override those access control mechanisms. Rather than disabling all TSF enforcement for a certain user ID, like on basic UNIX systems, (which can be simulated on the TOE as well) the TOE implements the concept of privileges and authorizations. As described in the TSS, privileges can be granted to processes that allow overriding limited aspects of access control enforcement – those privileges have been documented for the single access control security functions they relate to and in summary for the security function PV. In addition, trusted processes on the TOE enforce additional authorization checks as modeled in FDP_ACC.1(3) and FDP_ACF.1(3) and implemented in AZ – this allows to ensure that only users with certain authorizations can execute privileged programs. Roles as identified in FMT_SMR.1(1) are consequently implemented by means of authorizations assigned to users. For VIOS,

roles identified in FMT_SMR.1(2) are consequently implemented by means of user IDs, group IDs, and the role mechanism.

For AIX to enforce those mechanisms, the TOE needs to identify and authenticate users. This is implemented in IA as modeled by FIA_UID.2, FIA_UAU.2 and FIA_UAU.7, with strength requirements for authentication passwords in FIA_SOS.1. Security attributes are maintained for users as in FIA_ATD.1(1) and associated with processes acting on behalf of those users (FIA_USB.1(1)). Revocation of user attributes as part of the management functionality is modeled in FMT_REV.1(1) and (2).

For VIOS to enforce its mechanisms, it needs to identify and authenticate users. This is implemented in I&A as modeled by FIA_UID.2, FIA-UAU.2, and FIA_UAU.7, with strength requirements for authentication passwords in FIA_SOS.1. Security attributes are maintained for users as in FIA_ATD.1(2) and associated with processes acting on behalf of those users (FIA_USB.1(2)). Revocation of user attributes as part of the management functionality is modeled in FMT_REV.1(3).

Security function management is offered for the functionality described above by the security function SM, as modeled in FMT_MSA.1(1) through FMT_MSA.1(5), FMT_MSA.3(1) through FMT_MSA.3(5), FMT_MTD.1(4), (6), and called out in FMT_SMF.1.

Auditing for security-related events is modeled in FAU_GEN.1 and FAU_GEN.2 and implemented in AU, with management and protection functionality for audit trails in FAU_SAR.1, FAU_SAR.2, FAU_SAR.3 and FAU_STG.1, FAU_STG.3 and FAU_STG.4 – protection of audit trails is implemented based on the access control and privilege mechanisms described above. Additional management functionality for auditing is provided, specifically in FMT_MTD.1(1), (2) and (3), FAU_SEL.1 and implemented as described in SM. A reliable time source for audit record generation is provided as in FPT_STM.1.

The VIOS access control is defined by the VIOS access control policy. Where the discretionary access control policy is for TOE users and objects, the VIOS access control policy is for VIOS specific objects; thus, FDP_ACC.1(4) and FDP_ACF.1(4) were added to make this distinction. Furthermore, the VIOS subsystem is not part of the LSPP, thus, separate SFRs were added to make this distinction clearer. The security management SFRs FMT_MSA.1(6) and FMT_MSA.3(6) were added to handle the dependencies for managing the VIOS access control policy. Since the roles are different from LSPP, the FMT_SMR.1(2) security functional requirement was added to satisfy the role requirements. FMT_SMF.1 was modified to include VIOS security management functions and FMT_MTD.1(7) was added to describe the management functions.

Object reuse is prevented for shared resources as specified in FDP_RIP.2, Note1, and FDP_RIP.3-AIX and implemented in OR.

Protection of the TSF is implemented by means of abstract machine testing and domain separation in TP, as modeled in FPT_AMT.1 and FPT_SEP.1, and by implementation of a central reference monitor meeting the requirements of FPT_RVM.1. RVM.2-AIX provides an additional protection against the possible malicious misuse of a common programming problem which can allow the security functions to be bypassed.

8.4.2 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.

8.4.3 Strength of function

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

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

9 Abbreviations

ACE	Access Control Entry
ACL	Access Control List
AIX	Advanced Interactive Executive
ANSI	American National Standards Institute
API	Application Programming Interface
ASN	Advanced Secure Networking
CC	Common Criteria
CC	Common Criteria for Information Technology Security Evaluation
CDE	Common Desktop Environment
CDE	Computer Desktop Environment
CIPSO	Common IP Security Options
CM	Configuration Management
DAC	Discretionary Access Control
DAC	Discretionary Access Control
EAL	Evaluation Assurance Level
EGID	Effective Group ID
EOF	End of File
EUID	Effective User ID
FIFO	First In First Out
FIPS	Federal Information Processing Standard
FPR	Floating Point Register
FSO	File System Object
FSP	Functional Specification
FTP	File Transfer Protocol
GA	General Availability
GID	Group ID
GPR	General Purpose Register
HLD	High Level Design
HTML	Hypertext Markup Language
I&A	Identification and Authentication
ID	Identification
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
IP	Internet Protocol
IPC	Internet Protocol Control56
ISO	International Standards Organization
ISSO	Information System Security Officer
JFS	Journalled File System
KAT	Kernel Authorization Table
LFS	Logical File System
LPP	Licensed Product Package
LSPP	Labeled Security Protection Profile
MAC	Mandatory Access Control

NFS	Network File System
NIM	Network Install Manager
OID	Object Identification
OR	Observation Report
PDF	Portable Data Format
PP	Protection Profile
PP	Protection Profile
RPC	Remote Procedure Call
RPC	Remote Procedure Call
RSH	Remote Shell
SA	System Administrator
SAR	Security Assurance Requirement
SED	Stack Execution Disable
SEM	Superuser Emulation Mode
SFP	Security Function Policy
SFR	Security Functional Requirement
SL	Sensitivity Label
SMIT	System Management Interface Tool
SO	System Operator
ST	Security Target
ST	Security Target
TCB	Trusted Computing Base
TCP	Transmission Control Protocol
TOE	Target of Evaluation
TOE	Target of Evaluation
TSC	TSF Scope of Control
TSF	TOE Security Functions
TSF	TOE Security Function
TSP	TOE Security Policy
UDP	User Datagram Protocol
UID	User ID
VFS	Virtual File System
VIOS	Virtual I/O Server
VMM	Virtual Memory Manager