AIX 5L Version 5.2
Maintenance level 5200-06
Security Target

Version: 1.02
Status: Released
Last Update: 2005-10-14

Document History

Version	Date	Changes	Summary	Author
0.10	2005-01-07	All chapters	Changes from AIX 5.2 B to AIX 5.2 I, based on ST version 1.11 from 2003-03-28 (previous authors: Julie Haugh, Andreas Siegert). Introduced Disk Erase functionality as part of the TSF.	David Ochel, atsec
0.11	2005-01-07	All chapters	Changed ALC_FLR.1 to ALC_FLR.3, corrected targeted maintenance level.	D. Ochel
0.12	2005-01-10	All chapters	Incorporated review comments	D. Ochel
0.13	2005-01-13	All chapters	Moved to MS Office Format, minor typos fixed.	Andreas Siegert, atsec
0.14	2005-03-14	All chapters	Incorporated evaluator's comments; moved CC to version 2.2; corrected several spelling and wording issues; clarified evaluated configuration in 2.4; rephrased environmental threats in 3.2.2; updated HDD erase; updated description of assurance measures in 6.4.	D. Ochel
0.15	2005-03-22	All chapters	Specified Program Number for AIX; added UDFS for DVD-ROM support; specified hardware platforms; added minor clarifications; added SOF-claim for HDD erase.	D. Ochel
0.16	2005-03-22	Chapter 3	Added note on threat agents to P.ERASE; small formatting and wording changes.	D. Ochel
0.17	2005-03-23	Chapter 4,5	Added SOF for disk erasure	A. Siegert
0.18	2005-03-29	All chapters	Re-established formatting of SFR operations; addressed evaluator comments; streamlined section numbering in chpt. 5; corrected identification of hardware platforms; minor editorial changes.	D. Ochel
0.20	2005-04-01	FIA_USB.1	Editorial changes to clarify wording.	D. Ochel
0.30	2005-05-12	Various	Updated audited events in FAU_GEN.1; provided clarification on login mechanisms, semaphores and TSF databases in TSS; updated P.ERASE and SOF claims to clarify that erased HDDs stay within TSC.	D.Ochel, A. Siegert
1.0	2005-07-28	Chapter 2	Necessary PTFs	A. Siegert
1.01	2005-08-17	Chapter 2.3	Removed Adobe LPP	A. Siegert
1.02	2005-10-14	Various	Consistently introduced UDFS (DVD-ROM support) throughout the ST; several editorial changes.	D. Ochel

© IBM 2002, 2003, 2005

Table of Contents

© IBM 2002, 2003, 2005

© IBM 2002, 2003, 2005

© IBM 2002, 2003, 2005

© IBM 2002, 2003, 2005

References

[CC] Common Criteria for Information Technology Security Evaluation,
January 2004, Version 2.2, Part 1-3, CCIMB-2004-01-001, CCIMB-2004-01-002, CCIMB-2004-01-003.

[CEM] Common Methodology for Information Technology Security Evaluation, January 2004, Version 2.2,
CCIMB-2004-01-004.

[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

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

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

© IBM 2002, 2003, 2005

1 Introduction

This is the Security Target for the evaluation of AIX 5L Version 5.2 Maintenance level 5200-06, also known as AIX 5.2 I.

1.1 ST Identification

Title: AIX 5L Version 5.2 Maintenance level 5200-06 Security Target AIX 5.2 I, Version 1.02

Keywords: AIX, AIX 5L, AIX 5.2, general-purpose operating system, POSIX, UNIX, LPAR

This document is the Security Target for the CC evaluation of the operating system IBM AIX 5L for POWER V5.2, Program Number 5765-E62, with Recommended Maintenance Package 5200-06,(hereafter: AIX 5.2 I).

1.2 ST Overview

This Security Target documents the security characteristics of the AIX 5.2 I operating system.

AIX 5.2 I is a highly-configurable UNIX-based operating system which has been developed to meet the requirements of the Controlled Access Protection Profile developed by the Information Systems Security Organization within the National Security Agency to map the TCSEC C2 class of the U.S. Department of 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.

Several servers running AIX 5.2 I can be connected to form a distributed system. The communication aspects within AIX 5.2 I 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 Controlled Access Protection Profile version 1.d [CAPP].

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

Note: The evaluation assurance level named in the Protection Profile is EAL 3 with no augmentation. This Security Target claims an evaluation assurance level of EAL 4 augmented by ALC_FLR.3. Since EAL 4 is hierarchical to EAL 3 conformance to the assurance requirements of the Protection Profile is given.

In addition this Security Target has replaced the security functional requirements FIA_UAU.1 and FIA_UID.1 as listed in the Protection Profile by the security functional requirements FIA_UAU.2 and FIA_UID.2 which are hierarchical to the ones listed in the Protection Profile. Compliance to the security functional requirements as listed in the Protection Profile is therefore still given.

After the CAPP has been published and evaluated, AIS 32, Final Interpretation 065 of the CC has been published and subsequently integrated into the current version of the CC, which defines a new family has been added to the FMT class. This family FMT_SMF has just one component FMT_SMF.1. AIS 32, Final Interpretation 065 also has modified the definition of the components FMT_MOF.1, FMT_MSA.1 and FMT_MTD.1 such that they now contain a dependency on the new component FMT_SMF.1. To comply with this interpretation and satisfy the dependencies of FMT_MSA.1 and FMT_MTD.1, FMT_SMF.1 has been added as a functional requirement. This then is additional to the requirements defined in the CAPP. Furthermore, functionality to erase hard disk drives has been added to this Security Target.

EAL 4 has been augmented by ALC_FLR.3 since this is also covered by the Mutual Recognition Arrangement.

1.4 Strength of Function

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

© IBM 2002, 2003, 2005

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 [CAPP].

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.

Administrative User: This term refers to an administrator of an AIX system. Some administrative tasks requires use of the *root* username and password so that they can become the superuser (with a user ID of 0) 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), window system objects and miscellaneous objects. See section 6.2.5 on object reuse for a list of all objects handled by AIX 5.2 I.

Product: The term product is used to define all hardware and software components that comprise the distributed AIX 5.2 I system.

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

Role: A role represents a set of actions that an authorized user, upon assuming the role, can perform.

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 5.2 I 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.

© IBM 2002, 2003, 2005

2 TOE Description

The target of evaluation (TOE) is the operating system AIX Version 5.2 I.

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.

The AIX 5.2 I evaluation shall consist of a distributed, closed network of high-end, mid-range and low-end IBM pSeries servers running the evaluated version of AIX 5.2 I. The hardware platforms selected for the evaluation shall consist of machines which are projected to be available when the evaluation has completed and to remain available for a substantial period of time afterwards.

The TOE Security Functions (TSF) consists of those parts of 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.

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

The TOE shall include installation from CDROM and the network.

The TOE shall include standard networking applications, such as ftp, rlogin, rsh and NFS. Port-filtering will be used to protect network applications which might otherwise have security exposures.

The TOE shall include the X-Window graphical interface and many X-Window applications. 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 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 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 role.

Although AIX supports a concept of privileges that allows to define several roles with specific administrative rights, this Security Target does not include those privileges as part of the security requirements and security enforcing functions. Instead the standard Unix model of unprivileged users and a system administrator with full root privileges is used. So, whenever this Security Target mentions the system administrator role it is identical to the term „root”.

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.

© IBM 2002, 2003, 2005

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.

AIX 5.2 I is the platform for a large amount of commercial and scientific applications.

AIX 5.2 I 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 5.2 I has significant security extensions compared to standard UNIX systems:

- Access Control Lists,
- Integrity Protection,
- A Journaled File System,
- Integrated login framework.

AIX 5.2 I provides easy to use interfaces for users and system administrators:

- SMIT for system and user administration.

2.2 Summary of Security Features

The primary security features of the product are:

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

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

2.2.1 Identification and Authentication

AIX 5.2 I provides identification and authentication based upon user passwords. The quality of the passwords used can be enforced through configuration options controlled by AIX 5.2 I. Other authentication methods (e. g. Kerberos authentication) that are supported by AIX 5.2 I 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.

2.2.2 Auditing

AIX 5.2 I can collect extensive auditing information about security related actions taken or attempted by users, ensuring that users are accountable for their actions.

© IBM 2002, 2003, 2005

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, and other modified information.

This audit trail can be analyzed to identify attempts to compromise security and determine the extent of the compromise.

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. In addition, AIX Supports ACLs on Sockets for TCP connections.

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 5.2 I 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.

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.

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

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

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

2.3 Software

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

© IBM 2002, 2003, 2005

- IBM AIX 5L for POWER V5.2, Program Number 5765-E62, with Recommended Maintenance Package 5200-06.

The TOE documentation is supplied on CD-ROM.

The following table contains a list of LPPs / File Sets that make up the TOE.

Table 2-2: List of LPPs / File Sets

LPP Name	Description
Java	Various libraries, commands and classes associated with Java.
Mozilla	Mozilla browser and other client applications
X11	X Windows server, libraries and applications
bos	AIX Base Operating System
devices	AIX supported devices
sysmgt	System management tools.

The following list of PTFs is required on the evaluated system:

PTF	LPP
U802353	bos.mp64 5.2.0.62
U802354	bos.mp 5.2.0.62
U802365	bos.rte.security 5.2.0.63

2.4 Configurations

The evaluated configurations are defined as follows:

- The CC Evaluated file set must be selected at install time
- If a windowing environment is to be used, the CDE file set must be selected at install time.
- The role based system administration features of AIX 5.2 I are not included.
- AIX 5.2 I supports the use of IPv4 and IPv6, only IPv4 is included.
- 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.
- Dynamic Partitioning (Dynamic LPAR, DLPAR) is not supported in the evaluated configuration, i.e. the dynamic (de-) allocation of resources to a partition during operations is not allowed and must be prevented by organizational means in the IT environment.

The TOE comprises one or more of the server machines (and optional peripherals) listed in section 2.4.2 running the system software listed in table 2-2 (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.

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.

© IBM 2002, 2003, 2005

2.4.1 File systems

The following file system types are supported:

- the AIX journaled file system, jfs2,
- the standard remote file system access protocol, nfs (V3);
- the High Sierra file system 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 DAC attributes inferred from the underlying process’ DAC attributes.

2.4.2 Technical Environment for Use

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

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

- IBM pSeries Symmetric Multiprocessor (SMP) Systems, using Power5 CPUs (p5 520, p5 570, p5 595)

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

- all terminals and printers supported by the TOE
- all storage devices and backup devices supported by the TOE (hard disks, CD- and DVD-ROM drives, streamer drives, floppy disk drives)1 - note that the Erase Disk functionality supports SCSI hard disk drives only
- all printer devices supported by the TOE

c) Network connectors supported by the TOE (e.g. Ethernet, Token Ring, etc.) supporting TCP/IP services over the TCP/IP protocol stack.

2.4.2.1 LPAR Environment

The logical partitioning capable pSeries eServers that represent the underlying hardware for the TOE support a logical partitioned environment that enables the pSeries systems to run multiple logical partitions concurrently. In a logical partition, an operating system instance runs with dedicated resources: processors, memory, and I/O slots. These resources are statically assigned to the logical partition. The total amount of assignable resources is limited by the physically installed resources in the system. Because the implementation of logical partitioning is static, one has to shut down every operating system instance in all logical partitions to change the resource assignment of running logical partitions.

From a functional point of view, applications on top of an operating system are running inside partitions in the same way they run on a stand-alone pSeries machine. There are no issues when moving an application from a stand-alone server to a partition. Operating system software needs to be modified in some areas to call Hypervisor functions instead of native code. The design of partitioning-capable pSeries servers is such that one partition is isolated from software running in the other partitions, including protection against natural software defects and even deliberate software attempts to break the partition barriers.

The logical resources of the underlying hardware that can be assigned to a partition are:

- Processors
- Main memory regions
- I/O slots

---

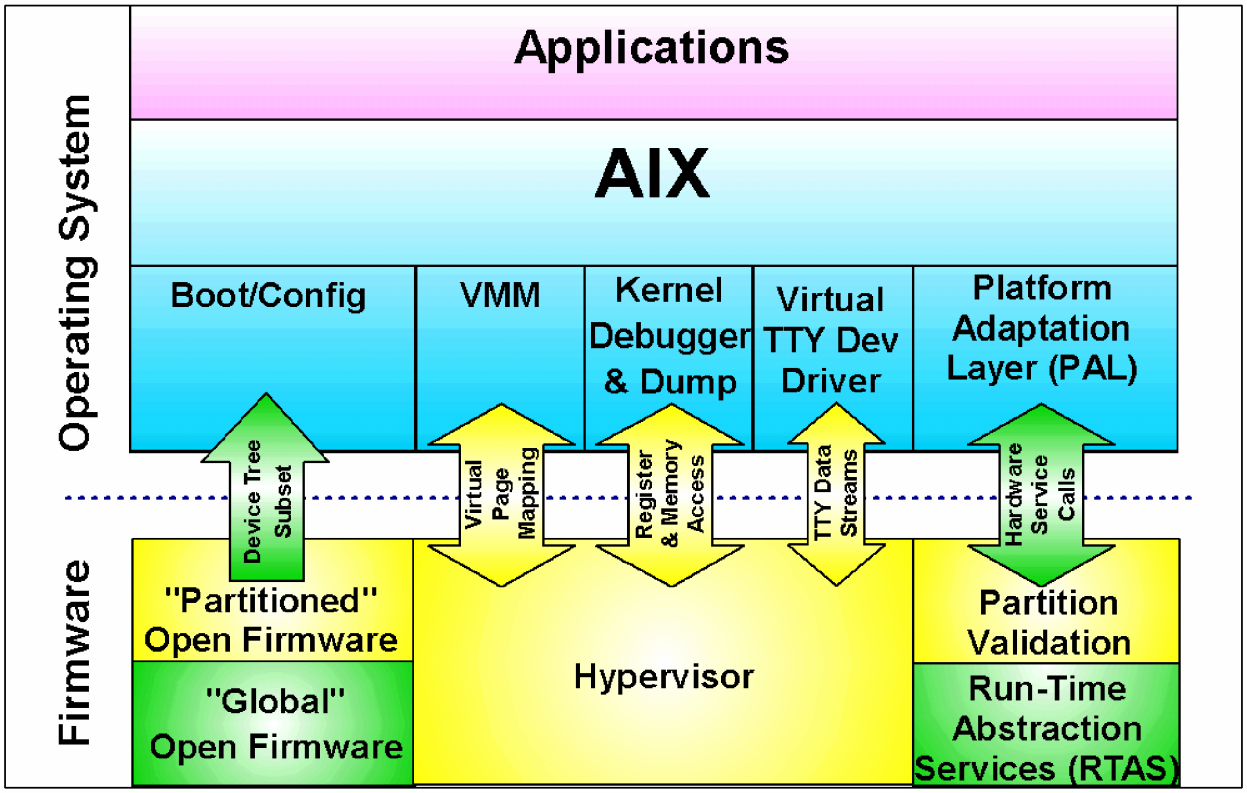
[1] The system distinguishes between storage and backup devices. Storage devices are hardware devices holding parts of the AIX file system, such as hard disks and CD ROMs. Backup devices are devices used for archiving data like floppy disks and streamer tapes that do not have a file system. Note that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

© IBM 2002, 2003, 2005

The assignment of those resources to the individual logical partitions is stored in non-volatile RAM. This part of the NVRAM is maintained by a “Service Processor” and can not be read or modified directly by the TOE running in a logical partition. The assignment itself is performed by a System Administrator, who uses a “Hardware Management Console” (HMC) to define those assignments. The HMC communicates with the service processor that accepts the commands from the HMC and sets the values to define the logical partitions in the non-volatile RAM (NVRAM) accordingly. A Run-Time Abstraction Layer (RTAS) provides an abstraction mechanism for platform service calls.

The functions of the underlying LPAR architecture need to be used by different parts of the TOE. The following figure shows the parts of AIX that interact with the functions of the IT environment. Adaptations in AIX have been made to enable the TOE to interact in an LPAR specifíc way with the VMM, virtual TTY console, RTAS and kernel debugger.

Please note that the support of static logical partitions does not introduce any additional security functionality for the TOE - the separation between partitions and protection of the TOE from operating systems running in other logical partitions on the same underlying machine is completely enforced by the underyling machine.



© IBM 2002, 2003, 2005

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 with.

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.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.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.

3.2.2 Threats to be countered by measures within the TOE environment

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

© IBM 2002, 2003, 2005

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.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.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.AUTHORIZED_USERS	Only those users who have been authorized to access the information within the system may access the system.
P.NEED_TO_KNOW	The right to access specific data objects is determined on the basis of: a) the owner of the object; and b) the identity of the subject attempting the access; and c) the implicit and explicit access rights to the object granted to the subject by the object owner.
P.ACCOUNTABLE	The users of the system shall be held accountable for their actions within the system.
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.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.

3.4 Assumptions

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

© IBM 2002, 2003, 2005

3.4.1 Physical Aspects

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

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

3.4.2 Personnel Aspects

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

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

A.COOP Authorized users possess the necessary authorization to access at least some of the information man-aged by the TOE and are expected to act in a cooperating manner in a benign environment.

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

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

3.4.3 Connectivity Aspects

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

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

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

© IBM 2002, 2003, 2005

4 Security Objectives

4.1 Security Objectives for the TOE

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

O.DISCRETIONARY_ACCESS The TSF must control 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.AUDITING The TSF must record the security relevant actions of users of the TOE. The TSF must present this information to authorized administrators.

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

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

O.ERASE The TOE shall offer a mechanism to overwrite user-accessible blocks of SCSI hard disk drives with pre-defined bit patterns.

4.2 Security Objectives for the TOE Environment

All security requirements listed in this section are targetted 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 clearance of the system.

The media on which authentication data is stored must not be physically removable from the distributed system by unauthorized users.

Users must not disclose their passwords to other individuals.

OE.INSTALL Those responsible for the TOE must establish and implement procedures to ensure that the hardware, software and firmware components that comprise the distributed system are distributed, installed and configured in a secure manner. This includes the static configuration of logical partitions of the LPAR feature of the TOE environment.

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

OE.INFO_PROTECT Those responsible for the TOE must establish and implement procedures to ensure that information is protected in an appropriate manner. In particular:

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

© IBM 2002, 2003, 2005

	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.
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.RECOVER	Those responsible for the TOE must ensure that procedures and/or mechanisms are provided to assure that, after system failure or other discontinuity, recovery without a protection (i.e., security) compromise is obtained.
OE.SOFTWARE_IN	Those responsible for the TOE shall ensure that the system shall be configured so that only an administrator can introduce new trusted software into the system.
OE.SERIAL_LOGIN	Those responsible for the TOE shall implement procedures to ensure that users clear the screen before logging off where serial login devices (e.g. IBM 3151 terminals) are used.

The following security objective applies in environments where specific threats to distributed 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.
OE.HW_SEP	The underlying hardware must provide separation mechanism that can be used by the TOE to protect the TSF and TSF data from unauthorized access and modification.

The following security objective applies 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.

© IBM 2002, 2003, 2005

5 Security Requirements

5.1 TOE Security Functional Requirements

5.1.1 Requirements Taken from Protection Profile(s)

The security functional requirements for the TOE are as defined in [CAPP] with all the operations performed to SFRs that have been left to the security target author. 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). In addition the security functional requirement FMT_SMF.1 has been added, as well as the iteration FDP_RIP.3-AIX addressing the Erase Disk functionality offered by the TOE.

The security functional requirements of the Protection Profile are listed together with the Application Notes and Rationales given in the Protection Profile. For some security functional requirements a specific Application Note for AIX has been added.

5.1.2 Extended Component Definition

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.

b) 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.

5.2 Security Audit (FAU)

5.2.1 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 **5-1** (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/NOTE4

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

© IBM 2002, 2003, 2005

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

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

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

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

Table 5-1: Auditable Events

Section	Component	Event	Details (Event Names)
5.2.1	FAU_GEN.1	Start-up and shutdown of the audit functions.	**start_up: AUD_It (cmd=1)** **shutdown: AUD_It (cmd=4)**
5.2.2	FAU_GEN.2	None	
5.2.3	FAU_SAR.1	Reading of information from the audit records.	**See below**[1]
5.2.4	FAU_SAR.2	Unsuccessful attempts to read information from the audit records.	**FILE_Open**[2]
5.2.5	FAU_SAR.3	None	
5.2.6	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**
5.2.7	FAU_STG.2	None	
5.2.8	FAU_STG.3	Actions taken due to exceeding of a threshold.	**AUD_Lost_Recs**
5.2.9	FAU_STG.4	Actions taken due to the audit storage failure.	**AUD_Lost_Recs**
5.3.1	FDP_ACC.1	None	
5.3.2	FDP_ACF.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**

[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

[2] This requirement was added in the CAPP 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.

© IBM 2002, 2003, 2005

Section	Component	Event	Details (Event Names)
			MSG_Mode **MSG_Delete** **MSG_Create** **SEM_Owner** **SEM_Create** **SEM_Delete** **SEM_Mode** **SHM_Create** **SHM_Delete** **SHM_Owner** **SHM_Mode**
5.3.3	FDP_RIP.2	None	
5.3.4	Note 1	None	
5.3.5	FDP_RIP.3-AIX	None	
5.4.1	FIA_ATD.1	None	
5.4.2	FIA_SOS.1	Rejection or acceptance by the TSF of any tested secret.	**USER_Login** **PASSWORD_Change** **USER_SU**
5.4.3	FIA_UAU.**2**	All use of the authentication mechanism.	**USER_Login** **PASSWORD_Change** **USER_SU**
5.4.4	FIA_UAU.7	None	
5.4.5	FIA_UID.**2**	All use of the user identification mechanism, including the identity provided during successful attempts.	**USER_Login** **PASSWORD_Change** **USER_SU**
5.4.6	FIA_USB.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**
5.5.1	FMT_MSA.1	All modifications of the values of security attributes.	**PROC_Environ** **PROC_Privilege** **PROC_Execute** **PROC_RealUID** **PROC_AuditID** **PROC_SetUserIDs** **PROC_RealGID** **PROC_SetGroups**
5.5.2	FMT_MSA.3	Modifications of the default setting of permissive or restrictive rules. All modifications of the initial value of security attributes.	**Object auditing events defined for /etc/security/user**
5.5.3	FMT_MTD.1	All modifications to the values of TSF data.	**Object auditing events for the TSF files containing the TSF data**
5.5.4	FMT_MTD.1	All modifications to the values of TSF data.	**PROC_Sysconfig** **AUD_it** **AUD_Bin_Def**
5.5.5	FMT_MTD.1	All modifications to the values of TSF data.	**USER_Change**
5.5.6	FMT_MTD.1	All modifications to the values of TSF data.	**USER_Change** **PASSWORD_Change** **PASSWORD_Flags**
5.5.7	FMT_REV.1	All attempts to revoke security attributes.	**USER_Change**
5.5.8	FMT_REV.1	All modifications to the values of TSF data.	**FILE_Acl**

© IBM 2002, 2003, 2005

Section	Component	Event	Details (Event Names)
5.5.9	FMT_SMR.1	Modifications to the group of users that are part of a role.	**USER_Change**
5.5.9	FMT_SMR.1	Every use of the rights of a role. (Additional / Detailed)	**PROC_Privilege**
5.6.1	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)**
5.6.2	FPT_RVM.1	None	
5.6.3	FPT_SEP.1	None	
5.6.4	FPT_STM.1	Changes to the time.	**PROC_Adjtime**

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 Discretionary Access Control such that only the System Administrator can access the file. When an installation requires to audit the use of the audit command, object auditing can be used where the diag command file is the object and execution is the access mode.

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

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

Application Note 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.

5.2.3 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 PP: 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.

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

Application Note for AIX: The access control to audit files within AIX is regulated by the discretionary access control of AIX. 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.

5.2.4 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

© IBM 2002, 2003, 2005

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

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

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.

5.2.5 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) **the following audit record fields:**

- **audit event**
- **user's login name**
- **event status**
- **time the record was written**
- **command name**
- **process ID**
- **ID of the parent process**
- **kernel thread ID**
- **name of the host that generated the audit event**

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

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

Application Note for AIX: AIX provides two commands for audit data processing: AUDITPR and AUDITSELECT. Details are described in chapter 6.

5.2.6 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) **file name**

c) **event type**

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

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

Application Note 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.

© IBM 2002, 2003, 2005

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

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

Application Note from AIX: Protection of stored audit records from deletion and modifications is performed using the discretionary access control mechanisms of AIX.

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

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

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

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

Application Note from AIX: AIX 5.2 I has been modified to implement such a variable and have it configurable.

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

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

Application Note for AIX: In the case all audit bins are full, AIX 5.2 I 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 necessry actions to make disk space available for auditing.

5.3 User Data Protection (FDP)

5.3.1 Discretionary Access Control Policy (FDP_ACC.1)

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

© IBM 2002, 2003, 2005

Application Note from PP: For most systems there is only one type of subject, usually called a process or task, which needs to be specified in the ST.

Named objects are those objects which are used to share information among subjects acting on the behalf of different users, and for which access to the object can be specified by a name or other identity. Any object that meets this criterion but is not controlled by the DAC policy must be justified.

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

Rationale from PP: This component supports the O.DISCRETIONARY_ACCESS objective by specifying the scope of control for the DAC policy.

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

5.3.2 Discretionary Access Control Functions (FDP_ACF.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:

permission bits and extended permission. (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 Entries).

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

IPC objects:

permission bits

Access rights for IPC objects are:
- read
- write

TCP ports:

Access control lists with entries of the following form:

user@host

user@subnet

group@host

group@subnet

The only access right is the right to set up a connection on the specified port

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:

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

© IBM 2002, 2003, 2005

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.

IPC objects:

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

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

the group id of the current process is equal to the goup 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

TCP ports:

Setting up a connection from another system to a TCP port on a given system can be regulated by access control lists. A connection can only be established to a TCP port if the connection is coming from a user and a host where the ACL for the TCP port has entry either of the form user@host, group@host, user@subnet or group@subnet matches the userid or groupid and the host or subnet.

In addition ports with numbers larger than 1024 can be turned into privileged ports, i. e. a local user that wants to start a server process listening on such a port must have root privileges.

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

File System Objects:

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

TCP ports:

Services listed in the file /etc/security/services are exempt from ACL checks

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

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

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

Rationale from PP: This component supports the O.DISCRETIONARY_ACCESS objective by defining the rules which will be enforced by the TSF.

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

© IBM 2002, 2003, 2005

5.3.3 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 PP: 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.

Rationale from PP: This component supports the O.RESIDUAL_INFO objective.

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

5.3.4 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.3

Application Note from PP: 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.

Rationale from PP: This component supports the O.RESIDUAL_INFO objective.

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

5.3.5 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.

5.4 Identification and Authentication (FIA)

5.4.1 User Attribute Definition (FIA_ATD.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) Security-relevant Roles; and

e) **Audit Classes**.

Application Note from PP: 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.

[3]This requirement was added in the CAPP Protection Profile to address resources that are not directly allocated to objects. Chapter 6 explains in detail how object resuse is handled for many kind of resources, also those that are not objects defined in this Security Target

© IBM 2002, 2003, 2005

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

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

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

Application Note for AIX: The only security relevant role defined within the TOE is the role of a system administrator. This role is identified by the userid of zero. The system needs to be configured such that access to files for system administration is restricted such that no user not having a userid of 0 can have access to those critical files.

5.4.2 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 PP: The method of authentication is unspecified by the CAPP, but must be specified in a ST. The method which is used must be shown to have low probability that authentication data can be 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)

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

Application Note for AIX: The TOE supports a number of configuration parameter 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.

5.4.3 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 PP: 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.

Rationale from PP: This component supports the O.AUTHORIZATION objective by specifying what actions unauthenticated users may perform.

Application Note for AIX: AIX does not allow any TSF mediated action of a user that is not authenticated. The Controlled Access Protection Profile specifies FIA_UAU.1 which allows to define actions that a user may perform before being authenticated. Since FIA_UAU.2 is hierarchical to FIA_UAU.1, conformance to the Protection Profile is achieved.

5.4.4 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

© IBM 2002, 2003, 2005

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

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

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

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

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

Rationale from PP: This component supports the O.AUTHORIZATION objective by specifying what actions unidentified users may perform.

Application Note for AIX: AIX does not allow any TSF mediated action of a user that is not identified. The Controlled Access Protection Profile specifies FIA_UID.1 which allows to define 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.

5.4.6 User-Subject Binding (FIA_USB.1)

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

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) **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:

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 a user:

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, so all audit records can be traced to the real user that executes the program.

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, so all audit records can be traced to the real user that executes the program.

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

© IBM 2002, 2003, 2005

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

Application Note: 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.

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

5.5 Security Management (FMT)

5.5.1 Management of Object Security Attributes (FMT_MSA.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 **system administrators and the owner of the object. In the case of TCP ports modification to access control lists is restricted to system administrators only.** FMT_MSA.1.1

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

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

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

5.5.2 Static Attribute Initialization (FMT_MSA.3)

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 allow the **administrators and the owner of the object** to specify alternative initial values to override the default values when an object or information is created. FMT_MSA.3.2

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

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

5.5.3 Management of the Audit Trail (FMT_MTD.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 PP: 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.

© IBM 2002, 2003, 2005

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

5.5.4 Management of Audited Events (FMT_MTD.1)

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

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

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

5.5.5 Management of User Attributes (FMT_MTD.1)

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

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

5.5.6 Management of Authentication Data (FMT_MTD.1)

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

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

a) authorized administrators; and

b) users authorized to modify their own authentication data

Application Note from PP: 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 re-authentication of the requester's identity at the time of the request.

Rationale from PP: This component supports the O.AUTHORIZATION and O.MANAGE objectives by ensuring integrity and confidentiality of authentication data.

5.5.7 Revocation of User Attributes (FMT_REV.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

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

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

Application Note from PP: 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

© IBM 2002, 2003, 2005

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. In a distributed environment the developer must provide a description of how the "immediate" aspect of this requirement is met.

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

Application Note for AIX: The immediate revocation method that can be used in AIX 5.2 I 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.

5.5.8 Revocation of Object Attributes (FMT_REV.1)

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

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; and

b) **Access rights to file system and IPC objects are checked when the object is opened. Revocations of access rights for file system objects become effective the next time a user affected by the revocation tries to open a file system object.**

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

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

Application Note for AIX: Immediate revocation for file system objects is not implemented in AIX 5.2 I. AIX 5.2 I uses delayed revocation as described in the application note from the PP. Immediate revocation is partly used within the NFS file system, which is described below in the TOE summary specification.

5.5.9 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**

Application Note for AIX: This security functional requirement has been added to the CC Version 2.2 as a result of AIS 32, Final Interpretation 065. This security functional requirement is not included in the CAPP, because the CAPP was developed before CCIMB 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 and CC Version 2.2.

5.5.10 Security Management Roles (FMT_SMR.1)

The TSF shall maintain the roles: FMT_SMR.1.1

a) authorized administrator;

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

© IBM 2002, 2003, 2005

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

d) **No other roles**

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

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

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

Rationale from PP: This component supports the O.MANAGE objective.

5.6 Protection of the TOE Security Functions (FPT)

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

Rationale from PP: This component supports the O.ENFORCEMENT objective by demonstrating that the underlying mechanisms are working as expected.

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

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

Rationale from PP: This component supports O.ENFORCEMENT objective by ensuring that the TSP is not being bypassed.

5.6.3 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 PP: 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.

© IBM 2002, 2003, 2005

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

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

Rationale from PP: This component supports the O.AUDITING objective by ensuring that accountability information is accurate.

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 to exactly terminate the 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.

5.7 Strength of Function

The claimed minimum strength of function is SOF-medium.

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

5.8 TOE Security Assurance Requirements

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

5.9 Security Requirements for the IT Environment

The following requirements are stated on the underlying processor, that has to provide the mechanism to protect the TSF and TSF data from unauthorized access and tampering. This is expressed with the following security functional requirement for the processor used to execute TOE software:

5.9.1 FDP_ACC.1 Subset access control

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

5.9.2 FDP_ACF.1 Security attribute based access control

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

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

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

© IBM 2002, 2003, 2005

FDP_ACF.1.3	The TSF shall explicitly authorise access of subjects to objects based on the following additional rules: **some dedicated processor registers may be read but not modified when the instruction accessing the register is in user mode**.
FDP_ACF.1.4	The TSF shall explicitly deny access of subjects to objects based on the **following rule: none**.

5.9.3 FMT_MSA.3 Static attribute initialization

FMT_MSA.3.1	The TSF shall enforce the **memory access control policy** to provide **permissive** default values for security attributes that are used to enforce the SFP.
FMT_MSA.3.2	The TSF shall allow the **no role** to specify alternative initial values to override the default values when an object or information is created.
Application Note:	The „default" values in this case are seen as the values the processor has after start-up. They have to be „permissive", sine 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.

The following requirements target the operation of the TOE in an LPAR-enabled environment, where actually more than one logical partition is configured on the underlying machine the TOE runs on:

5.9.4 FDP_ACC.1 (LPAR) Subset access control

FDP_ACC.1.1	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**.

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

FDP_ACF.1.1	The TSF shall enforce the **LPAR resource access control policy** to objects based on **the partition number**.
FDP_ACF.1.2	The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed: **a 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.1.3	The TSF shall explicitly authorize access of subjects to objects based on the following additional rules: **none**.
FDP_ACF.1.4	The TSF shall explicitly deny access of subjects to objects based on the **no other rules**.

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

© IBM 2002, 2003, 2005

5.10 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.

© IBM 2002, 2003, 2005

6 TOE Summary Specification

6.1 Security Enforcing Components Overview

6.1.1 Introduction

AIX 5.2 I 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 5.2 I 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 (i.e., via the exec system call), or create a copy of itself (i.e., 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.

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 root identity.

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

© IBM 2002, 2003, 2005

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. The following types of file systems are included in the evaluated configuration: Journaled File System 2 (JFS2), CDROM File System (CDRFS), DVD-ROM 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 attributes inferred from the underlying process' DAC attributes.

6.1.3 Non-Kernel TSF Services

The non-kernel TSF services are:

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

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 host computer in the system 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.

© IBM 2002, 2003, 2005

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). Non-TSF processes may communicate with other hosts in the distributed system using the TOE's implementation of those protocols.

The higher-level network services are built on TCP or UDP. While the TOE supports the TCP application protocols listed below, only the timed application protocol uses UDP and is provided by the TOE in the evaluated configuration. The application protocols provided using TCP are:

- 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.

AIX 5.2 I includes multiple X Windows clients in addition to an X Windows server on each host. Each server accepts connections from local clients using UNIX domain sockets.

6.1.5 Security Policy Overview

Since the TOE is distributed across multiple host computers, each running a semiautonomous instance the AIX operating system, the policy is described as follows:

- There is not a single kernel; rather, there is an AIX kernel running on each host computer in the system.
- The system does not have a common memory space; rather, each host in the system has its own memory space. Memory management, segmentation and paging are all managed locally, without respect to other hosts.
- Identification and authentication (I&A) is performed locally by each host computer, but uses a common database. 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).
- Neither the process ID, nor the associated thread IDs, are unique within the system; rather, a PID, and its associated TIDs, are unique on each host within the system. Process and thread management is performed locally, without respect to other hosts.
- 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 performed locally by each of the host computers and 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 Access Control List (ACL). The extended permissions mechanism is a significant enhancement beyond traditional UNIX systems, and permits control of access based on lists of users and/or groups to whom specific permissions may be individually granted or denied. For TCP based services, an additional type of ACLs is provided that can be used to restrict user access to specific services.
- Object reuse is performed locally, without respect to other hosts.
- Audit is performed locally by each host computer. 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). Audit tools are available to merge audit files from the various hosts.
- Interrupt handling is performed locally, without respect to other hosts.

© IBM 2002, 2003, 2005

- Privilege is based on the root identity. All privileged processes (setuid root programs and programs run under the root identity) start as processes with all privileges enabled. Unprivileged processes, which include setgid trusted processes, start and end with no privileges enabled.

6.1.6 TSF Structure

The TSF is the portion of the system that is responsible for enforcing the system's security policy. The TSF of AIX 5.2 I are distributed across each pSeries host computer and 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 AIX 5.2 I Distributed System 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, scripts, shells, and standard UNIX utilities that run with administrative privilege, as a consequence of being invoked by a user with the root identity. Non-kernel TSF software also includes daemons that provide system services, such as networking and managing audit data, as well as setuid and setgid 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 5.2 I Distributed 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 peform security functions and therefore present part of the TSF interface.

© IBM 2002, 2003, 2005

- Distributed Services, (e.g. telnet, NFS, rsh) 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 within the system. At a lower level, it allows a host on the distributed system to request a specific service from another host within the system on behalf of a user. Examples of requested services include, reading data from a designated file (i.e. NFS), executing a command line (e.g. rsh) or transfering whole files (e.g. FTP). At the lowest level, it allows a subject on one host in the system to request a connection (i.e. TCP), or deliver data (i.e. UDP) to a listening subject. 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 5.2 I are the same as the interfaces for ordinary users; the administrator logs into the system with a standard, untrusted, identity and password, and after assuming the root identity uses standard AIX commands to perform administrative tasks.

The system is composed of one or more pSeries computer systems. Each of these host computers may be in one of the following states: shut down, initialization, single-user mode, or multi-user secure state. Administration entails the configuration of multiple computers and the interactions of those computers, as well as the administration of users, groups, files, printers, and other resources within the system.

AIX 5.2 I 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 5.2 I Distributed 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 graphical 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.

The system maintains an administrative database on a Network File System (NFS) server, referred to as the administrative master server. The remaining hosts import the administrative data from the master server through ordinary NFS client operations.

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 administartive 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 5.2 I Distributed System is defined as a host's entry into multi-user mode with auditing fully operational, and with the administrative databases NFS-mounted from the master server. At this point, the host accepts user logins and services network requests across the distributed system. If these facilities are not available, the host is considered to be in a non-secure state. Although it may be operational in a limited sense and available for an administrative user to perform system repair, maintenance, and diagnostic activity, the TSF are not in full operation and is not necessarily protecting all system resources according to the security policy.

With respect to auditing this Security Target does not define a minimum level of events that need to be audited. But is 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.

© IBM 2002, 2003, 2005

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:

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 5.2 I Distributed 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. All those services use a common mechanism for authentication described as function IA.2 in this section. They all use the administrative database described in function IA.1 in this section. The administrative database is managed by system administrators but users are allowed to modify their own password using the passwd command. Function IA.3 describes the authentication process for those network services that require authentication. Function IA.4 describes the change of the user's identity using the su command. Function IA.5 described the login process when a user logs in at a terminal. Function IA.6 describes the logoff process.

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

To ensure that the AIX 5.2 I Distributed System is administered as a single entity, the system maintains a single administrative database on a Network File System (NFS) server, referred to as the administrative master server. The remaining hosts import the administrative data from the master server through ordinary NFS client operations.

The system maintains a consistent administrative database by making all administrative changes only on the designated NFS server and exporting the database files to all the other computers in the system. A user ID on any computer refers to the same individual on all other computers. In addition, the password configuration, name-to-UID mappings, and other data are identical on all hosts in the distributed system.

Administrators, through the SMIT administrative interface, 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 setuid program with the owning userid of 0. 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.

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

The file */etc/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.

© IBM 2002, 2003, 2005

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 minumum 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 such that the passwords satisfy the requirements defined in FIA_SOS.1.

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.

The mounting of the master happens during the AIX 5.2 I Distributed System boot sequence, and the system is not in the secure state until a successful mount of the administrative data occurs. If the mount fails, the system is inaccessible as it hangs on the mount and will remain so until the administrator corrects the problem on the host causing the mount to fail. Once the problem is resolved and the mount can occur successfully, the system will complete booting.

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

6.2.2.2 Common Authentication Mechanism (IA.2)

AIX 5.2 I 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 and groups)

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

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

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 for interactive login sessions on the console of a user's local host;
- the *Telnet* protocol and the *rlogin* protocol for ordinary interactive login sessions on any host in the system;
- the *rsh*, *RCP* and the *rexec* protocol for remote shell, copy, and single command executions;
- the *FTP* protocol for interactive file transfer;
- the *xlock* program that is used to lock active X window sessions
- and the *su* command for changing user identity during a session

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.

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

Note: xlock is not a full login mechanism but uses the same authentication mechanism to re-authenticate a user who has locked an X window session.

© IBM 2002, 2003, 2005

6.2.2.3.1 The Login Program

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

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. User is changed to his or her home directory
8. 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 of any host in the distributed system, access to other hosts within the system may occur through one of six network protocols: telnet, rlogin, rsh, rcp, rexec, and FTP.

6.2.2.3.2.1 *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.

6.2.2.3.2.2 *Login with rlogin*

The rlogin protocol includes user identity as part of the protocol information passed from host to host. User is not permitted to switch identity between hosts using -l option. See the description of rsh command execution below for details on the enforcement mechanism.

6.2.2.3.2.3 *Command execution using rsh*

The rsh protocol includes user identity as part of the protocol information passed from host to host. User is not permitted to switch identity between hosts using -l option. rshd checks to see that the remote and local user names are the same. Remember that remote user ID information flows with the rsh connection request as part of the protocol. The requirement that a privileged/reserved port be used as part of the setup insures that the information in the protocol flow was created by a trusted process.

6.2.2.3.2.4 *Command execution using rcp*

The RCP protocol includes user identity as part of the protocol information passed from host to host. User is not permitted to switch identity between hosts using -l option. See the description of command execution using rsh for details of the enforcement mechanism.

6.2.2.3.2.5 *Command execution using rexec*

© IBM 2002, 2003, 2005

The rexec protocol always requires the user to enter a valid user identity and password. The authentication is performed by invoking the common authentication mechanism directly rather than by invoking login. User can change identity if password is known.

6.2.2.3.2.6 *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.

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

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 the AIX 5.2 I Distributed System is to allow appropriately authorized individuals the ability to assume the root identity. In this system 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 identity.

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

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

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 attached terminal, the TSM program changes the ownership of */dev/lft0, /dev/kbd0*, */dev/mouse0* 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, mouse, and graphics adapter. At system initialization, */dev/lft0* grabs the keyboard, mouse and graphics adapter devices.

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. 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 */dev/mouse0* device contains only one channel, which is grabbed by the */dev/lft0* device on system startup. Attempts to open additional instances of the */dev/mouse0* device will result in an error message.

The login process executes a revoke to invalidate any open file descriptors for */dev/lft0* 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.

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

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/lft0*, */dev/kbd0*, */dev/mouse0*, and */dev/rcm0* is returned to root when the logoff occurs.

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

© IBM 2002, 2003, 2005

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.

Audit data is generated separately on each host in the distributed system and may be managed and analyzed either separately on each host in the distributed system, or merged and analyzed on a single system. AIX includes tools for pre-selecting and post-selecting audit data, viewing audit trails, and merging multiple audit trails into one file.

Function AU.1 discusses the format of the audit record. Here the data common to all audit records is explained.

Function AU.2 explains the system configuration files that control the audit feature of AIX.

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 6-1: 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.	

The above listed table shows that for each audit record the information required by FAU_GEN.1 and FAU_GEN.2 is contained in the audit record.

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.

© IBM 2002, 2003, 2005

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.

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

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 processors 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 6-2: 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, 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.

© IBM 2002, 2003, 2005

Field	Definition
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.

Auditselect allows to selectively extract individual audit records as required by FAU_SAR.1 and FAU_SAR.3.

6.2.3.4 Audit Review (AU.4)

Three different commands exist for the review of audit records in the distributed system: *auditpr, auditmerge* 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 6-3. Available Fields From *auditpr*. These fields are 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 6.2.3.4, 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 in the system, 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 format as required by FAU_SAR.1.

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

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 AIX 5.2 I Security Guide instructs the administrator that if the cached and permanent audit trails are kept other than in the /audit directory, then the alternate directory must be protected from access by non-root users.

This satisfies the requirements FAU_SAR.2, FAU_STG.1 and FMT_MTD.1 „Audit Trail".

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

© IBM 2002, 2003, 2005

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 5.2 I 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 5.2 I 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 5.2 I 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. If a host halts because it cannot collect audit records, the other hosts in the distributed system are not affected, unless the host is acting as the administrative master. The AIX 5.2 I 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 5.2 I 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.

This methods prohibits the loss of audit data as required by FAU_STG.3 and FAU_STG.4.

6.2.4 Discretionary Access Control (DA)

This section outlines the general DAC policy in AIX 5.2 I as implemented for resources where access is controlled by permission bits and optionally, extended permissions; principally these are the objects in the file system. In all cases the policy is based on user identity (and in some cases on group membership associated with the user identity). To allow for enforcement of the DAC policy, all users must be identified and their identities authenticated.

Details of the specific DAC policy applied to each type of resource are covered in the section "Discretionary Access Control: File System Objects" and the section "Discretionary Access Control: IPC Objects".

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

Finally, a subject with an effective UID of 0 is exempt from all restrictions and can perform any action desired, including the execution of files for which at least one exec DAC bit is set.

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 5.2 I. DAC is implemented with permission bits and, when specified, extended permissions.

© IBM 2002, 2003, 2005

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.

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

Permission bits are the standard UNIX DAC mechanism and are used on all AIX 5.2 I 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 mechanism is supported only for file system objects and provides a finer level of granularity than do permission bits.

6.2.4.1 Permission Bits (DA.1)

AIX 5.2 I 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 is not the root identity, then the access attempt is denied.
- If the process is the root identity, and the attempted access is an execution of the object, the access is granted only if at least one of the execution bits is set and otherwise denied.

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

6.2.4.2 Extended Permissions (DA.2)

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.

© IBM 2002, 2003, 2005

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 Access Control List (ACL). ACLs have a textual representation (used with commands such as ACLGET) and binary representations (for storage in the file system).

Together the Discretionary Access Control functions implement the requirements defined in FDP_ACC.1 and FDP_ACF.1. Details of the Discretionary Access Control mechanism for individual object types are described below.

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.

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

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*

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.

6.2.4.3.1.2 *DAC Attributes 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 the root identity.
- 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 is the root identity.
- The owning user ID of an object may only be changed by the root identity.

© IBM 2002, 2003, 2005

6.2.4.3.1.3 *DAC 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 (i.e., 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).

6.2.4.3.1.4 *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 (i.e., 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 5.2 I Security Guide.

Applications that want to revoke tty access of other processes can use the revoke() and frevoke() system calls to revoke all current access of other processes, forcing them to reopen the file and to undergo the associated access checks again.

6.2.4.3.2 DAC: Ordinary File

In addition to read and write, ordinary files support the execute permission. 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. One has to keep in mind that AIX doesn't support set-UID or set-GID scripts.

6.2.4.3.3 DAC: Directory

The execute permission bit for directories governs the ability to name 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.

© IBM 2002, 2003, 2005

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

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 permission bits and extended permissions can be used. 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 extended permissions 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. 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.

6.2.4.3.7 DAC: Special Cases for NFS File Systems

An NFS filesystem may contain any of the supported object types of the underlying filesystem. That includes device special files. Non-regular files or directories which occur on NFS filesystems are treated similar to objects defined on the local filesystem -- a device special file on an NFS mounted filesystem will reference the underlying device on the local system. It would not reference the device on the remote system.

DAC checks by the NFS server for file contents permit a subject with the same effective owning user ID as the file to have access to the contents regardless of the DAC attributes. This is used to support the standard UNIX semantics for access to open files, because such access is not re-validated when a file's DAC attributes change. This special case relies on the property that, ordinarily, only a file's owner changes its DAC while the file is open, and it is thus sufficient to handle the owner specially.

DAC changes do have immediate effect for users other than the owner, unlike local files: if an NFS-accessed file's DAC is changed to deny access, any subsequent read or write operation to an open file will fail if the operation would no longer be permitted by the new DAC attributes.

However, this can never grant additional access, because the client would have checked the access when the file was opened and not permitted more access than the DAC attributes allowed at open time.

The filesystem maintains a "handle" on the credentials which were used at the time an NFS file was opened. It is those credentials which are used to reference files via NFS, not the current process credentials which might be modified by setuid().

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

6.2.4.4 Discretionary Access Control: TCP Connections (DA.4)

TCP based services can be protected with ACLs as well. By specifying port, host/network,user combinations, ports can be restricted to specific hosts and/or users. For example specifying port 6000, machine colorado and user joe, only this user coming from machine colorado will be able to connect to the X server. The remote hosts use TCP AH headers to send the information about the user together with the connection request. AIX 5.2 I checks */etc/security/acl* for permitted clients.

With the DACinet Feature of AIX 5.2 I the concept of privileged ports (ports that can only be opened by the superuser, typically all ports below 1024) is extended so that any port now can be a privileged port. A bitmap of privileged ports is defined to hold information on whether a port is privileged. A system administrator can modify this bitmap.

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

© IBM 2002, 2003, 2005

6.2.4.5 Discretionary Access Control: IPC Objects (DA.5)

6.2.4.5.1 DAC: Shared Memory

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

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. Since a SMS exists only within a single host in the distributed system, rebooting the particular host where the SMS is present is sufficient to revoke all access to that SMS.

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.

Access permissions can be changed by any process with an effective UID equal to the owning UID or creating UID of the SMS. Access permissions can also be changed by any process with an effective UID of 0, also known as running with the root identity.

6.2.4.5.2 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 (i.e., 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.

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 with an effective UID of 0.

© IBM 2002, 2003, 2005

6.2.4.5.3 DAC: Semaphores

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

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

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

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

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

Access permissions can be changed by any process with an effective UID equal to the owning UID or creating UID of the semaphore. Access permissions can also be changed by any process with an effective UID of 0.

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

Note: In addition to the regular IPC semaphores, AIX also supports memory mapped semaphores that are accessible within the memory mapped address space of processes that can share the mapped address space. There is no explict access control on these semaphores (handled by the msem_init, msem_lock, msem_unlock, msem_remove, msleep and mwakeup subroutines). The mmap routines map files into shared memory and all access control is performed via the DAC protection mechanisms of the mapped files or memory.

6.2.5 Object Reuse (OR)

Object Reuse is the mechanism that protects against scavenging, or being able to read information that is left over from a previous subject's actions. Four general techniques are applied to meet this requirement in the AIX 5.2 I Distributed System: explicit initialization of resources on initial allocation or creation, explicit clearing of resources on release or de-allocation, management of storage for resources that grow dynamically, and administrator-initiated wiping of hard disk drives

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 5.2 I 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.

Hard disk wiping provides means to an administrator to overwrite information on hard disks with bit patterns in order to render previously stored information on the disks unrecoverable. Rather of being a system property, this is a function that can be invoked by the administrator.

© IBM 2002, 2003, 2005

The following sections describe in detail how object reuse is handled for the different types of objects and data areas and how the requirements defined in FDP_RIP.2 and FDP_RIP.3-AIX are satisfied.

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

All file system objects (FSOs) available to general users are accessed by a common mechanism for allocating disk storage and a common mechanism for paging data to and from disk. This includes the Journaled File System (JFS2) and Network File System (which exists physically as a JFS2 volume on a server host). It includes both normal and large JFS2 file systems.

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

For this analysis, the term FSO refers not only to named file system objects (files, directories, device special files, named pipes, and UNIX domain sockets) but also to 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.

6.2.5.1.1 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 is seeked past the EOF and then written, this leaves a hole in the file that will subsequently be read as zeroes.

6.2.5.1.2 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 i-node 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 i-node 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.

6.2.5.1.3 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.

© IBM 2002, 2003, 2005

6.2.5.1.4 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 filesystem blocks, unless the device references a filesystem (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).

6.2.5.1.5 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.

6.2.5.1.6 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.

6.2.5.1.7 Object Reuse: Socket Special File (UNIX Domain)

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

This function contributes to satisfy the security requirement FDP_RIP.2 and „Subject Residual Information Protection" (Note 1).

6.2.5.2 Object Reuse: IPC Objects (OR.2)

AIX 5.2 I 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.

This function contributes to satisfy the security requirement FDP_RIP.2 and „Subject Residual Information Protection" (Note 1).

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

6.2.5.3.1 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.

This function contributes to satisfy the security requirement FDP_RIP.2 and „Subject Residual Information Protection" (Note 1).

6.2.5.4 Object Reuse: Miscellaneous Objects (OR.4)

6.2.5.4.1 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.

© IBM 2002, 2003, 2005

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.

This function contributes to satisfy the security requirement FDP_RIP.2 and „Subject Residual Information Protection" (Note 1).

6.2.5.5 Object Reuse: Hard disk drives (OR.5)

For SCSI disks that do not participate as "pdisks" in RAID arrays, the diagnostic subystem (cf. section 6.2.7.7) 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.

This function satisfies the security requirement FDP_RIP.3-AIX.

6.2.6 Security Management (SM)

This section describes the functions for the management of security attributes that exist within AIX 5.2 I.

6.2.6.1 Roles (SM.1)

The configuration of AIX 5.2 I that is subject to this evaluation does not use the full roles and privilege concepts of AIX. Instead a simplified role model is used., that just supports two roles: administrators and normal users.

In the evaluated configuration a user has the role of an administrator when he is allowed to *su* to root. Root itself will not be used as a userid where a user can directly log in to. So every administrator has his/her own user ID, which is used to log into the system and which will appear in all audit records produced by this user. This allows to trace also administrator activities to individual users.

6.2.6.1.1 Administrators

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

6.2.6.1.2 Normal Users

Normal users can not perform actions that require administrator privileges. They can only execute those setuid root programs they have access to. 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.

Note: AIX 5.2 I supports a finer role concept but this is not part of the evaluated configuration. In the evaluated configuration only users and administrators exist where administrators operate with root privileges.

© IBM 2002, 2003, 2005

6.2.6.2 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 6-4. 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 indicidual 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*

This allows to audit all the events defined in FAU_GEN.1. 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.

© IBM 2002, 2003, 2005

This contributes to satisfy the requirements of FAU_GEN.1, FAU_SEL.1, FMT_MTD.1 „Audit Events", FMT_MTD.1 „Audit Trail" and FMT_SMF.1.

A description of the structure of those files and the syntax of the entries can be found in *AIX 5.2 I Files Reference* document.

6.2.6.3 Access Control Configuration and Management (SM.3)

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

Permissions can 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.

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

6.2.6.4 Management of User, Group and Authentication Data (SM.4)

6.2.6.4.1 Creating new Users

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

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
- 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 file */etc/security/user*, /etc/passwd and /etc/group.

6.2.6.4.2 Modification of user attributes

User attributes can be modified by the system administrator. Modifications of user attributes require the modification of the administration database that contains the user attributes (mainly */etc/security/user*).

6.2.6.4.3 Management of Authentication Data

The system administrator has the capability to define rules and restrictions for passwords used to authenticate users. The parameter 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.

© IBM 2002, 2003, 2005

minother	Minimum number of non-alphabetic characters the new password must contain. (Other characters are any ASCII printable characters that are non-alphabetic 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.

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

6.2.6.5 Time Management (SM.5)

AIX 5.2 I provides the standard Unix functions to manage the system clock. The time can be set or modified by an administrator. 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.

This satisfied the requirements of FPT_STM.1.

6.2.7 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 5.2 I. The memory and process management components of the kernel ensure a user process cannot access kernel storage or storage belonging to other processes.

Non-kernel TSF software and data are protected by DAC and process isolation mechanisms. In the evaluated configuration, the reserved user ID root, or other reserved IDs equivalent to root, owns TSF directories and files In general, files and directories containing internal TSF data (e.g., audit files, batch job queues) are also protected from reading by DAC 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 host in the distributed system is adequately protected. A description of the boot logical volume can be found in section 5.3.16, Initialization and Shutdown.

6.2.7.1 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.

© IBM 2002, 2003, 2005

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

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

6.2.7.2 Kernel (TP.2)

The AIX 5.2 I 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 5.2 I 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).

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

6.2.7.3 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., NFS), 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.

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

6.2.7.4 Trusted Processes (TP.4)

Trusted processes in AIX 5.2 I are processes running in user mode but with root 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

© IBM 2002, 2003, 2005

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 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 root 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 mechanism check, if the process has the required access rights for the attempted type of access.

Any program executed with root privileges has the ability to perform the actions of a trusted process. It is therefore important that a site operating 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. For identification and authentication they contribute to satisfy the security functional requirements FIA_UAU.2, FIA_UAU.7 and FIA_UID.2.

This function also contributes to FPT_SEP.1.

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

6.2.7.5 TSF Databases (TP.5)

Table 6-5 identifies the primary TSF databases used in AIX 5.2 I 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 the system administrator.

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.

When the system comprises more than one machine, the administrative databases are shared between the hosts. A single master copy of the databases is maintained on an administrative server and made available for NFS-mounting by all the hosts in the system. A particular user or group has the same identity on all hosts in the system. Some databases are maintained independently on each host in the system, while others are synchronized through sharing.

Table 6-5 . Administrative Databases. This table lists other administrative files used to configure the TSF.

Database	Purpose
/etc/ftpusers	Limits accees to FTP
/etc/group	Stores group names, supplemental GIDs, and group members for all system groups.
/etc/hosts	Contains hostnames and their address for hosts in the network. This file is used to resolve a hostname into an Internet address in the absence of a domain name server.
/etc/inetd.conf	Configures start of network daemons
/etc/inittab	Contains commands to srcmaster daemon that starts other system daemons.

© IBM 2002, 2003, 2005

Database	Purpose
/etc/passwd	Stores user names, UIDs, primary GID, home directories for all system users.
/etc/security/acl	Specification of TCP port, host (or subnet), and user/group at that host or subnet allowed access to the port.
/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/lastlog	Stores time/date of last successful and unsuccesful login attempts for each user. Stores the number of unsuccessful login attempts since the last successful one.
/etc/security/login.cfg	Defines attributes enforced when logging in or changing passwords
/etc/security/passwd	Defines user passwords in one-way encrypted form, plus additional characteristics including previous passwords, password quality parameters.
/etc/security/portlog	Records ports locked as a result of login failures
/etc/security/priv	Defines the privileged ports as part of the access control on TCP ports
/etc/security/services	Specification of service names to be used by DACINET in the style of /etc/services.
/etc/security/user	Defines supplementary data about users, including audit status, required password characteristics, access to su command.
/usr/lib/security/mkuser.default	Defaults for user account creation

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

6.2.7.6 Internal TOE Protection Mechanisms (TP.6)

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

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

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

6.2.7.7 Diagnosis (TP.7)

AIX 5.2 I 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.

This function contributes to satisfy the security requirement FPT_AMT.1. The error log file contributes also to FAU_GEN.1

© IBM 2002, 2003, 2005

6.3 Supporting functions not part of the TSF

6.3.1 System Management Tools

The evaluated configuration of AIX 5.2 I 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 5.2 I.

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 5.2 I 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 5.2 I.

Table 6-6 Mapping Assurance Requirements to Documentation

Assurance Component	Assurance measures
ACM_AUT.1	IBM uses automated CM tools with discretionary access control to provide configuration management for all evaluation evidence.
ACM_CAP.4	See above
ACM_SCP.2	IBM has a problem tracking procedure in place that covers all aspects of ACM_SCP.2.
ADO_DEL.2	IBM uses commercially established techniques for secure delivery of the TOE.
ADO_IGS.1	Installation, generation and start-up procedures are provided as part of the guidance, including a dedicated discussion of the specifics of the evaluated configuration.
ADV_FSP.2	IBM provides a functional specification.
ADV_HLD.2	The high level design is described in an overview document and subsystem specific documents.
ADV_IMP.1	The full source code of the parts of AIX that build the TSF is provided for

© IBM 2002, 2003, 2005

Assurance Component	Assurance measures
	the evaluation.
ADV_LLD.1	Low level design documentation is provided for all subsystems that implement TSF.
ADV_RCR.1	IBM provides correspondence analyses for all relevant design aspects.
ADV_SPM.1	A Security Policy Model is provided.
AGD_ADM.1	Administrator guidance is provided.
AGD_USR.1	User guidance is provided.
ALC_DVS.1	Security procedures for the development environment are applied and documented
ALC_FLR.3	IBM has a mature customer support interface in place, including first, second and third level support and sophisticated means to automatically inform subscribed customers about potential security flaws and workarounds and fixes available to prevent their exploitation.
ALC_LCD.1	The software development process implemented as well as the tools used for development are adequate and documented.
ALC_TAT.1	See above.
ATE_COV.2	Detailed test plans are produced to test the functions of AIX 5.2 I. Those test plan include an analysis of the test coverage, an analysis of the functional interfaces tested and an analysis of the testing against the high-level design.
ATE_DPT.1	See above
ATE_FUN.1	IBM performs and documents functional testing to reach adequate test coverage and depth.
ATE_IND.2	The evaluation facility will perform and document independent testing.
AVA_MSU.2	A misuse analysis for the guidance is provided..
AVA_SOF.1	A Strength of Function Analysis is provided.
AVA_VLA.2	IBM performs and documents regular vulnerability analysis.

6.5 TOE Security Functions requiring a Strength of Function

The TOE has one security function (IA.1) that is implemented by a probabilistic or permutational mechanism. This is the authentication mechanism that uses passwords for user authentication. The strength claimed for this function is SOF-medium.

As explained above the System Administrator has several options he can use to force the users to select passwords that are not easy to guess. In addition the TOE limit the number of attempts an attacker has to guess passwords in a try and error method.

The hard disk erasure functionality in OR.5, which is not per se probabilistic or permutational, can also be considered subject to a strength of function rating to the extent that it overwrites existing information on drives with a certain level of “noise” that makes it hard to retrieve the original information stored on the disk. However, since the hard disk drive does not leave the TSC and is only accessible via the TOE-provided interfaces, a SOF claim is not applicable for this mechanism.

© IBM 2002, 2003, 2005

7 Protection Profile Claims

7.1 PP Reference

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

Security functional requirements have been added to those defined in the CAPP. FMT_SMF.1 was added to resolve dependencies from FMT_MSA.1 and FMT_MTD.1 that did not exist at the time of the CAPP registration, and FDP_RIP.3-AIX provides additional functionality that does not interfere with the TSP defined in CAPP.

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

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

Two organizational security policies and threats have been added (the Protection Profile only defines policies). One assumption on the TOE environment (A.NET_COMP) has been added to reflect the distributed nature of the TOE.

The objective O.ERASE has been added for the TOE.

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

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

In addition the Security Target has added security requirements for the IT environment (the processor used) to define the requirement for the underlying processor to provide the functions to implement effective separation of the TSF from untrusted software. This includes the requirements FDP_ACC.1, FDP_ACF.1 and FMT_MSA.3 for the IT environment. As well, security requirements for the IT environment have been added to address the necessary protection of hardware resources assigned to the TOE against access by software running in other partitions when running in an LPAR environment (FDP_ACC.1 (LPAR) and FDP_ACF.1 (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.3. Since the Evaluation Assurance Levels in the Common Criteria define a hierarchy, all assurance requirements of the Protection Profile are included in this Security Target. ALC_FLR.3 which has been added to the assurance requirements defined in the CAPP has no dependency on any other security functional requirement or security assurance requirement and is therefore an augmentation that has no effect on the security functional requirements or security assurance requirements stated in the Protection Profile.

© IBM 2002, 2003, 2005

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 Controlled Access 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 [CAPP].

8.1 Rationale for additional Threats, Assumptions and Organizational Security Policies

The CAPP does not lists threats but only addresses the following policies:

- P.AUTHORIZED_USERS
- P.NEED_TO_KNOW
- P.ACCOUNTABILITY

Those policies are also listed in this Security Target.

In addition, a new policy has been added with respect to the LPAR environment:

- P.STATIC

And one policy has been introduced to address the Disk Erase functionality added to the TSF:

- P.ERASE

[CAPP] also lists the following assumptions:

- A.LOCATE
- A.PROTECT
- A.MANAGE
- A.NO_EVIl_ADM
- A.COOP
- A.PEER
- A.CONNECT

Two assumptions have been added, since they had also been part of previous Security Targets for AIX (ITSEC Security Target for AIX 4.3):

- A.UTRAIN
- A.UTRUST

Another assumption

- A.NET_COMP

has been added to reflect the distributed nature of the TOE.

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.UAUSER
- T.UAACCESS
- T.UAACTION

Threats countered by the TOE environment are:

© IBM 2002, 2003, 2005

- TE.HWMF
- TE.COR_FILE
- TE.HW_SEP
- TE.LPAR

8.1.1 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_COM 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.

8.1.2 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 inlcusion of threats is not a violation of the conformance claim to the Security Target.

Note: There are other examples of evaluations where the Security Targets claims conformance with [CAPP] and where threats have been defined.

8.1.3 Rationale for additional Organizational Security Policies

All policies from [CAPP] have been adopted by the ST. In addition, P.STATIC has been added to harmonize the use of LPAR-enabled hardware as underlying systems in the IT environment. P.ERASE addresses security functionality provided by AIX as a response to consumer requirements.

8.2 Security Objectives Rationale

The following tables provide a map<ping 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

Marked elements are stipulated by CAPP.

© IBM 2002, 2003, 2005

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

Objective	Threat / Policy
O.AUTHORIZATION	T.UAUSER, P.AUTHORIZED_USERS
O.DISCRETIONARY_ACCESS	T.UAACCESS, P.NEED_TO_KNOW
O.AUDITING	T.UAUSER, T.UAACTION, P.ACCOUNTABLE
O.RESIDUAL_INFO	P.NEED_TO_KNOW, T.UAACCESS
O.MANAGE	P.AUTHORIZED_USERS, P.NEED_TO_KNOW, P.ACCOUNTABLE, T.UAUSER, T.UAACTION
O.ENFORCEMENT	P.AUTHORIZED_USERS, P.NEED_TO_KNOW, P.ACCOUNTABLE
O.ERASE	P.ERASE

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

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

Table 8-3: Mapping threats to objectives

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

Table 8-4: Mapping Assumptions to Objectives

Assumption	Objective
A.LOCATE	OE.PHYSICAL
A.PROTECT	OE.INFO_PROTECT, OE.PHYSICAL
A.MANAGE	OE.ADMIN, OE.INSTALL
A.NO_EVIL_ADMIN	OE.ADMIN, OE.INSTALL
A.COOP	OE.CREDEN
A.NET_COMP	OE.PROTECT, OE.INSTALL
A.PEER	OE.INSTALL
A.CONNECT	OE.SERIAL_LOGIN, OE.PROTECT, OE.PHYSICAL
A.UTRAIN	OE.INFO_PROTECT

© IBM 2002, 2003, 2005

A.UTRUST	OE.INFO_PROTECT

Table 8-5: Mapping Policies to Objectives

Policy	Objective
P.AUTHORIZED_USERS	O.AUTHORIZATION, O.MANAGE, O.ENFORCEMENT
P.NEED_TO_KNOW	O.DISCRETIONARY_ACCESS, O.MANAGE, O.ENFORCEMENT, O.RESIDUAL_INFO, OE.SOFTWARE_IN
P.ACCOUNTABLE	O.AUDITING, O.MANAGE, O.ENFORCEMENT
P.STATIC	OE.INSTALL
P.ERASE	O.ERASE

8.2.2 Security Objectives Sufficiency

T.UAUSER: The threat of an attacker impersonating an authorized user 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. Together those objectives ensure that no unauthorized user can impersonate as an authorized user.

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_INFO ensures that an authorized user can not gain access to the information contained in a resource after the resource has been released to the system for reuse.

T.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.

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

TE.COR_FILE: The threat of undetected loss of integrity of security enforcing or relevant files of the TOE is diminished by OE.INSTALL requiring procedures for secure distribution, installation and configuration of systems thereby ensuring that the system has a secure initial state with the required protection of such files, OE.PROTECT requiring protection of transferred data in a distributed system 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.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.

© IBM 2002, 2003, 2005

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

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

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

A.MANAGE: The assumption on competent administrators is covered by OE.ADMIN requiring competent and trustworthy administrators and OE.INSTALL requiring procedures for secure distribution, installation and configuration of systems.

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

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

A.NET_COMP: The assumption on network components to not modify transmitted data is covered by the objective OE.PROTECT requiring procedures and/or mechanisms to ensure a safe data transfer between systems as well as OE.INSTALL requiring proper installation and configuration of all parts of the 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.CONNECT: The assumption on controlled access to peripheral devices and protected internal communication paths is covered by OE.SERIAL_LOGIN for the protection of attached serial login devices, OE.PROTECT for the protection of data transferred between workstations and OE.PHYSICAL requiring physical protection.

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

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

P.AUTHORIZED_USERS (see CAPP 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 this functions an O.ENFORCEMENT ensuring the correct invocation of the functions.

P.NEED_TO_KNOW (see CAPP 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_INFO ensuring that resources do not release such information during reuse and by OE.SOTWARE_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.ACCOUNTABLE (see CAPP 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.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.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.

© IBM 2002, 2003, 2005

8.3 Security Requirements Rationale

Since this Security Target does only identify two additional security functional requirements for the TOE to the SFRs already defined in the CAPP (except for FMT_SMF.1), the security requirements rationale is widely adopted from this Protection Profile by simple reference.

FMT_SMF.1 has been added to achieve compliance with AIS 32, Final Interpretation 065, which defines dependencies of FMT_MSA.1 and FMT_MTD.1 (which are included in this Security Target) to this new component. It was identified that FMT_SMF.1 should be included mentioning the areas of management listed in the instantiations of FMT_MSA.1 and FMT_MTD.1 in the Security Target.

FDP_RIP.3-AIX has been added to meet the objective O.ERASE by providing a mechanism to overwrite residual information on hard disk drives upon request of administrators.

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.

In addition, security requirements for the IT environment have been added to address the threat that arises from running the TOE in an environment that provides logical partitions to allow the processing of several operating systems on one machine. To such logical partitions, dedicated processors, memory ranges and I/O slots are assigned - to protect the TSF and TSF data, this requires means to prevent access to resources that are assigned to the TOE by other operating systems. This is expressed using FDP_ACC.1 (LPAR) and FDP_ACF.1 (LPAR) defining an appropriate access control policy.

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 Refinements Rationale

This section provides the rationale for the selections and refinements made in the security requirements section for the security requirements defined in the CAPP.

In FAU_GEN.1 the table was augmented with the names of the audit events as defined for AIX 5.2 I. The table shows that all events that the CAPP requires to be audited one or more audit events of AIX 5.2 I can be associated with. Also, FIA_UAU.1 and FIA_UID.1 were refined to meet the hierarchically superior FIA_UAU.2 and FIA_UID.2 selected in the ST.

In FAU_SAR.3 the refinement shows the additional fields of the audit records that can be used in the evaluation of the audit record database using the auditselect command.

In FAU_SEL.1 the capability of AIX 5.2 I to define auditable events not only based on the user id but also based on the object (in this case: files) and the based on the event type is specified and further explained in the security function AU.2

For FAU_STG.3 the threshold value is defined as an absolute value as defined as one possible option in the rationale of the CAPP.

In FAU_STG.4 the possibility of AIX 5.2 I to count the number of audit records lost for auditable events of an authorized administrator is expressed. Normally the system would be configured to stop when the audit trail is full (which is a hard, but acceptable way to prevent the auditable events).

In FDP_ACC.1 the different objects that AIX 5.2 I controls with a discretionary access control function are listed. Since the Controlled Access Protection Profile has only one instantiation of FDP_ACC.1, FDP_ACF.1 and FMT_MSA.1, it was decided to subsume all the discretionary access control sub-policies under a single instantiation of those security

© IBM 2002, 2003, 2005

functional requirements. Although it is agreed that this does not support readability of the Security Target, we view this approach as possible in the CC framework and compliance with the CAPP is maintained.

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

In FIA_ATD.1 the audit classes have been added as the only additional security attribute of users within the evaluated configuration of AIX 5.2 I. Other attributes as for example stored in the file /etc/security/users are not seen as security attributes.

In FIA_USB.1 the way how AIX 5.2 I associates the real, effective and audit user id is expressed. While the effective user id and group id can change as the result of a su command or a program with the setuid or setgid attribute set, the real and audit user id are maintained and allow to trace activities to the real user that originated them.

In FMT_REV.1 „Revocation of User Attributes" the delayed revocation method has been added, since this is the standard way AIX 5.2 I behaves. To get immediate revocation the administrator has to force the user to log off after he has made the modifications to the users attribute. According to the rationale in the Controlled Access Protection Profile this method is seen as appropriate to satisfy this requirement.

In FMT_REV.1 „Revocation of Object Attributes" the AIX 5.2 I implementation of delayed revocation is defined. This is in accordance with the rationale defined in the Controlled Access Protection Profile.

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

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

those aspects are listed in this security functional requirement.

FMT_SMR.1 defines no additional roles than those identified in the Protection Profile. Although AIX 5.2 I in general provides the capability for a more fine grained role model, the lack of guidance on the configuration and use of such an extended role model in the user documentation has led to the decision to exclude this capability from the evaluated version. This is compliant with the application note in the Protection Profile as well as with the handling of this aspect in previous ITSEC and TCSEC evaluations.

FPT_AMT.1 describes the capability of an authorized administrator to start a separate diagnostic system that tests the underlying hardware. It should be mentioned that some basic checks are included in the system's init process and are performed during start-up of the system. Those checks are part of the underlying firmware and not part of AIX 5.2 I.

8.3.2 Security Requirements Coverage

Section 7.2.2 of the Controlled Access 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.

This section therefore only discusses those security functional requirements that are different from the ones listed in the Protection Profile.

Security functional requirements FIA_UAU.1 and FIA_UID.1 have been replaced by FIA_UAU.2 and FIA_UID.2 which both are stronger requirements than those defined in the Protection Profile. FIA_UAU.1 and FIA_UID.1 are both mapped to the objectives O.AUTHORIZATION, which requires requires that only authorized users gain access to the TOE and its resources. While FIA_UAU.1 and FIA_UID.1 both allow to define actions a user can porform without being properly identified and authenticated, no action is allowed by FIA_UAU.2 and FIA_UID.2. Therefore FIA_UAU.2 and FIA_UID.2 are much more in line with the objective stated by O.AUTHORIZATION than the security functional requirements defined in the Protection Profile.

© IBM 2002, 2003, 2005

Security functional requirement FMT_SMF.1 has been included as an additional functional requirement to address the objective O.MANAGE. Security management within the TOE consists of the management of user attributes (addressed by FMT_MTD.1 „Management of User Attributes"), management of object security attributes (addressed by FMT_MSA.1), management of the audit trail (addressed by FMT_MTD.1 „Management of the Audit Trail"), management of the audit events (addressed by FMT_MTD.1 „Management of Audited Events") and management of authentication data (addressed by FMT_MTD.1 „Management of Authentication Data").

FDP_RIP.3-AIX addressed the objective O.ERASE by providing a mechanism to overwrite residual information on hard disk drives upon request of administrators.

In addition to the rationale provided in the Protection Profile the following table shows that each security functional requirement addresses at least one objective.

Table 8-6: 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	O.DISCRETIONARY_ACCESS
FDP_ACF.1	O.DISCRETIONARY_ACCESS
FDP_RIP.2	O.RESIDUAL_INFO
Note 1	O.RESIDUAL_INFO
FDP_RIP.3-AIX	O.ERASE
FIA_ATD.1	O.AUTHORIZATION, O.DISCRETIONARY_ACCESS
FIA_SOS.1	O.AUTHORIZATION
FIA_UAU.2	O.AUTHORIZATION
FIA_UAU.7	O.AUTHORIZATION
FIA_UID.2	O.AUTHORIZATION
FIA_USB.1	O.DISCRETIONARY_ACCESS, O.AUDITING
FMT_MSA.1	O.DISCRETIONARY_ACCESS
FMT_MSA.3	O.DISCRETIONARY_ACCESS
FMT_MTD.1 Audit Trail	O.AUDITING, O.MANAGE
FMT_MTD.1 Audit Events	O.AUDITING, O.MANAGE
FMT_MTD.1 User Attributes	O.MANAGE
FMT_MTD.1 Authen. Data	O.AUTHORIZATION, O.MANAGE
FMT_REV.1 User Attributes	O.DISCRETIONARY_ACCESS, O.MANAGE
FMT_REV.1 Obj. Attributes	O.DISCRETIONARY_ACCESS
FMT_SMF.1	O.MANAGE
FMT_SMR.1	O.MANAGE
FPT_AMT.1	O.ENFORCEMENT
FPT_RVM.1	O.ENFORCEMENT
FPT_SEP.1	O.ENFORCEMENT
FPT_STM.1	O.AUDITING

© IBM 2002, 2003, 2005

8.3.3 Rationale for Security Requirements for the IT environment

In addition to the requirements of [CAPP] this Security Target has added security requirements for the IT environment. The requirements FDP_ACC.1, FDP_ACF.1 and FMT_MSA.3 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.

These 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.

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.

The following table shows the mapping of the security functional requirements for the IT environment to the security objectives for the IT environment.

Table 8-7: 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

© IBM 2002, 2003, 2005

8.3.4 Justification of explicitly expressed security requirements

The explicit requirement Note 1 is adopted from the CAPP, the substantiation of the CAPP 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.

8.3.5 Security Requirements Sufficiency

The internal consistency of the security functional requirements and the complete coverage of the defined security objectives by security functional requirements is demonstrated in section 7.2 of the CAPP and applies for this Security Target. As stated earlier, FMT_SMF.1 explicitly fulfills dependencies that were already implicitly fulfilled in CAPP, as required by international interpretation, and FDP_RIP.3-AIX introduces self-sufficient functionality that does not interfere with the CAPP-defined SFRs.

8.3.6 Security Requirements Dependency Analysis

The only security functional requirement added to the ones defined in the CAPP is FMT_SMF.1 as defined in AIS 32, Final Interpretation 065. This component has no dependency. But since with AIS 32, Final Interpretation 065 dependencies of the components FMT_MSA.1 and FMT_MTD.1 to the new component FMT_SMF.1 have been introduced, FMT_SMF.1 has been added to this Security Target to resolve those dependencies.

FIA_UAU.1 and FIA_UID.1 as defined in the CAPP 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.

FDP_RIP.3-AIX as defined in the extended component definition does not have any dependencies on other SFRs.

Since no other additional security functional requirements to those defined in the CAPP Protection Profile 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 CAPP.

With respect to the security requirements for the IT environment the dependencies of FMT_MSA.3 on FMT_MSA.1 and FMT_SMR.1 are not resolved. This has been justified in the application note for FMT_MSA.3 in the definition of the security requirements for the IT environment as well as in the chapter „Justification for unresolved dependencies". The dependency of FDP_ACC.1 on FDP_ACF.1 is resolved as well as the dependency of FDP_ACF.1 on FDP_ACC.1 and FMT_MSA.3. The dependencies of FMT_MSA.3 on FMT_MSA.1 and FMT_SMR.1 are not resolved as stated earlier.

The dependency of FDP_ACF.1 (LPAR) for the IT environment on FMT_MSA.3 has not been resolved. This is left to the author of the ST for the LPAR mechanism: The dependency of the TOE on access control to its resource has been unambiguously expressed by the LPAR resource access control policy defined in FDP_ACC.1 (LPAR) and FDP_ACF.1 (LPAR) - the static attribute initialization for this policy is considered an implementation issue that needs not be defined within this ST and should be left to the ST defining the actual implementation. The mutual dependencies between FDP_ACC.1 (LPAR) and FDP_ACF.1 (LPAR) have been resolved.

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.3, 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.

© IBM 2002, 2003, 2005

Since EAL4 is hierarchical to EAL3, all dependencies of security functional requirements on assurance requirements listed in the Protection Profile are also resolved.

8.3.7 Justification of unresolved dependencies

As demonstrated in the dependency analysis above, all dependencies between security requirements are resolved since all the dependencies within the CAPP Protection Profile are resolved, 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.3 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.

8.3.8 Strength of function

This Security Target claims in compliance with CAPP a SOF rating SOF-medium. This claim applies for FIA_SOS.1, whereby CAPP 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.

8.3.9 Evaluation Assurance Level

The EAL defined in the CAPP is EAL3, as CAPP addresses a generalized environment with a moderate level of risk to the assets. This security target claims EAL4 augmented with ALC_FLR.3, 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 shows that the IT security functions, as specified in the TOE summary specification, meet all security functional requirements for the TOE and work together to satisfy the TOE security functional requirements.

Table 8-8: 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 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

© IBM 2002, 2003, 2005

SFR	security functions (TOE summary specification)
	enforced by the audit file protection of AU.5. (This function uses the access control functions for files as described in DA.3 to protect the audit files from unauthorized access. Management of file access rights is covered by SM.3 but all this is covered by AU.5).
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.
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	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, DA.4 for TCP connections and DA.5 for IPC objects.
FDP_ACF.1	The discretionary access control is realized as described above by DA.1, DA.2, DA.3, DA.4 and DA.5. There the individual mechanisms for access control depending on the object type are described in detail.
FDP_RIP.2	Object residual information protection is realized by security functions for object reuse on file system objects (OR.1), IPC objects (OR.2), queing 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	The hard disk erase functionality implemented in OR.5 is designed to meet the requirements on overwriting all currently user-accessible blocks of a SCSI hard disk drive upon request of the administrator.
FIA_ATD.1	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 funtion 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 initated 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 initated by a

© IBM 2002, 2003, 2005

SFR	security functions (TOE summary specification)
	trusted process. Trusted processes are described in TP.4.
FIA_USB.1	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	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), DA.4 (TCP connections) and DA.5 (IPC objects).
FMT_MSA.3	Restrictive default values for security attributes are defined for the objects when they are created. Default values can be defined by the system administrator for all object types and by the user for file system objects created under his control. (see above, i.e. SM.3, DA.3, DA.4, and DA.5). Some default values are defined in TSF databases as defined in TP.5.
FMT_MTD.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 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 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 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_REV.1 User Attributes	The revocation of user security attributes as required in FMT_REV.1 is realized by the user management funcitons of SM.4.
FMT_REV.1 Obj. Attributes	Revocation of object security attributes is realized by the access control configuration and management function SM.3.
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 (TCP connections), DA.5 (IPC objects). In addition SM.3 defines some management functions. 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	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.

© IBM 2002, 2003, 2005

SFR	security functions (TOE summary specification)
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_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.

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

CAPP contains a (quite short) justification why the requirements themselves are mutually supportive. Although the Controlled Access Protection Profile has been evaluated and therefore no additional justification for the mutual support of the security requirements is necessary (the additional requirement is justified by AIS 32 Final Interpretation 065), here are some additional arguments for mutual support of the security functional requirements:

The requirements for auditing (FAU_GEN.1, FAU_GEN.2, FAU_SAR.1, FAU_SAR.2 FAU_SAR.3, FAU_SEL.1, FAU_STG.2, FAU_STG.3 and FAU_STG.4) define the requirements for an audit system by defining the events the TOE should be able to audit with the relation to the other security functional requirements. The association of each event to the user's identity is consistent with the requirement for user identification and authentication (FIA_UID.2 and FIA_UAU.2), so FAU_GEN.2 has to basis to associate audit events with the user that caused the event. In addition the requirement in FAU_GEN.1 to record the time and date of the event needs to be based on a reliable time stamp as required by FPT_STM.1. FAU_SAR.1, FAU_SAR.2 and FAU_SAR.3 ensure that audit records can be reviewed on a useful basis. In addition FAU_SEL.1 enables an administrator to avoid to be flooded with audit data by enabling him to adapt the type of events that are actually audited to his needs.

Audit data only makes sense when the audit data is complete or when at least the administrator is warned before the TOE gets into a state where auditing is no longer possible. FAU_STG.2 to FAU_STG.4 address this issue.

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

Object reuse is a useful requirement to prohibit unwanted access to information via resources that have not been prepared for reuse. Since the TOE supports access control, object resuse makes sense. This is addressed in FDP_RIP.2. The requirement „Note 1" extends this to objects that are not directly related to objects covered by the discretionary access control policy (like main memory). In addition, FDP_RIP.3-AIX provides functionality to format hard disk drives in a way that is supposed to make residual information unavailable even when a resource is accessed by means other than those provided by the TOE.

Identification and authentication is handled by FIA_ATD.1, FIA_SOS.1 FIA_UAU.2, FIA_UAU.7 FIA_UID.2 and FIA_USB.1 in a fairly conventional way. FIA_USB describes the way the effective user ID and group ID can be changed. Since the real user ID is the one taken for audit events, any change of the effective user ID will not result in a false user ID in the audit records. Therefore FIA_USB is not contradicting FAU_GEN.2.

In the management section the requirements for the management of Audit Trail, Audited Events, User Attributes and Authentication Data has been separated in the protection profile. Since they are clearly separated, they are not contradicting each other.

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

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

© IBM 2002, 2003, 2005

FPT_AMT.1 allows the administrator to perform tests of the underlying hardware to verify its correct operation. This is not contradicting any other requirement and is useful to verify the correct operation.

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

FPT_STM.1 is required by the audit requirement FAU_GEN.1, which needs a reliable time to be included in the audit data. This allows the administrator to identify when an event has happened.

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

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 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.

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.

8.5 PP Claims Rationale

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

The additional security objectives for the TOE do not violate the objectives defined in CAPP. Objectives for the TOE environment have been added to this ST in addition to the ones contained in CAPP to allow a more distinguished description of the TOE environment - this does not impact the conformance of this ST to the PP.

All but two security functional requirements in this ST are inherited from the CAPP and the operations allowed / required by the PP are performed and indicated in bold letters. Two security functional components (FIA_UAU.1 and FIA_UID.1) have been replaced by hierarchical higher ones (FIA_UAU.2 and FIA_UID.2). In both cases the only difference is the fact that no interaction with the TOE is allowed without proper user identification and authentication. This does not modify any of the rationale provided in the PP. In addition FMT_SMF.1 has been added to comply with AIS 32 Final Interpretation 065, which defines dependencies of two security functional requirements (FMT_MSA.1 and FMT_MTD.1) included in the PP. To satisfy those requirements the new security functional component FMT_SMF.1 has been added to the Security Target (anticipating that this security functional requirement will be added in an update to the Controlled Access Protection Profile). Upon consumer request, the additional SFR FDP_RIP.3-AIX has been added to address functionality that does not interfere with the CAPP specified TSP.

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

© IBM 2002, 2003, 2005

9 Abbreviations

AIX	Advanced Interactive Executive
ANSI	Amerivan National Standards Institute
CAPP	Controlled Access Protection Profile
CC	Common Criteria
CDE	Common Desktop Environment
DAC	Discretionary Access Control
DACINET	Discretionary Access Control for Internet Services
EOF	End of File
FIPS	Federal Information Processing Standard
FPR	Floating Point Register
FSO	File System Object
FTP	File Transfer Protocol
GA	General Availability
GPR	General Purpose Register
HTML	Hypertext Markup Language
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
ISO	International Standards Organization
JFS	Journalled File System
LFS	Logical File System
LPAR	Logical Partitioning
LPP	Licensed Product Package
NFS	Network File System
NIM	Network Install Manager
PDF	Portable Data Format
PP	Protection Profile
RPC	Remote Procedure Call
RSH	Remote Shell
SMIT	System Management Interface Tool
ST	Security Target
TCP	Transmission Control Protocol
TOE	Target of Evaluation
TSF	TOE Security Functions
UDP	User Datagram Protocol
VFS	Virtual File System
VMM	Virtual Memory Manager

© IBM 2002, 2003, 2005