1.

Red Hat Enterprise Linux (RHEL) Advanced Server (AS) Version 3 Update 5 Running on Unisys ES7000 Hardware Security Target

Version 1.0
01/04/07

Prepared for:
Unisys Corporation

Unisys Way
Blue Bell, PA 19424

Prepared By:
Science Applications International Corporation

Common Criteria Testing Laboratory

7125 Columbia Gateway Drive, Suite 300
Columbia, MD 21046

LIST OF TABLES

1. Security Target Introduction

This section identifies the Security Target (ST) and Target of Evaluation (TOE) identification, ST conventions, ST conformance claims, and the ST organization. The TOE is Red Hat Enterprise Linux (RHEL) Advanced Server (AS) Version 3 Update 5 Running on Unisys ES7000 Hardware provided by Unisys Corporation.

RHEL AS is a commercial operating system product developed by Red Hat, Inc. It is a version of Linux that has been developed not only to serve as a fully capable operating system, but also to provide security for commercial environments.

The Unisys ES7000 hardware platforms (specifically, the Unisys ES7000-4xx-M2 series) are mainframes designed and developed by Unisys Corporation. Each of these machines is designed to support numerous 32- or 64-bit Intel microprocessors, respectively, as well as other supporting and peripheral devices (memory, disks, CD and floppy disk drives, network cards, and other I/O devices such as keyboard and mice).

Though the TOE includes both Red Hat and Unisys components, it is assembled, delivered, and supported by Unisys.

The Security Target contains the following additional sections:

- Section 2 – Target of Evaluation (TOE) Description

This section gives an overview of the TOE, describes the TOE in terms of its physical and logical boundaries, and states the scope of the TOE.

- Section 3 – TOE Security Environment

This section details the expectations of the environment, the threats that are countered by the TOE and IT environment, and the organizational policy that the TOE must fulfill.

- Section 4 – TOE Security Objectives

This section details the security objectives of the TOE and IT environment.

- Section 5 – IT Security Requirements

The section presents the security functional requirements (SFR) for the TOE and IT Environment that supports the TOE, and details the assurance requirements for EAL3 augmented with ALC_FLR.2.

- Section 6 – TOE Summary Specification

The section describes the security functions represented in the TOE that satisfy the security requirements.

- Section 7 – Protection Profile Claims

This section presents any protection profile claims.

- Section 8 – Rationale

This section closes the ST with the justifications of the security objectives, requirements and TOE summary specifications as to their consistency, completeness, and suitability.

1.1 Security Target, TOE and CC Identification

ST Title – Red Hat Enterprise Linux (RHEL) Advanced Server (AS) Version 3 Update 5 Running on Unisys ES7000 Hardware Security Target

ST Version – Version 1.0

ST Date – 01/04/07

TOE Identification – Red Hat Enterprise Linux (RHEL) Advanced Server (AS) Version 3 Update 5 Running on Unisys ES7000 Hardware (specifically, the Unisys ES7000-4XX-M2)

TOE Developer – Red Hat, Inc. (for the operating system software) and Unisys Corporation (for the hardware)

Evaluation Sponsor – Unisys Corporation

CC Identification – Common Criteria for Information Technology Security Evaluation, Version 2.2, Revision 256, January 2004.

1.2 Conformance Claims

This TOE is conformant to the following CC specifications:

- Controlled Access Protection Profile (CAPP) Version 1.d, 8 October 1999
- Common Criteria for Information Technology Security Evaluation Part 2: Security Functional Requirements, Version 2.2, Revision 256, January 2004.
    - Part 2 Conformant
- Common Criteria for Information Technology Security Evaluation Part 3: Security Assurance Requirements, Version 2.2, Revision 256, January 2004.
    - Part 3 Conformant
    - Assurance Level: EAL 3 augmented with ALC_FLR.2
    - Strength of Function Claim: SOF-Medium

1.3 Conventions

This section specifies the formatting information used in the Security Target.

The following conventions have been applied in this document:

- Security Functional Requirements – Part 2 of the CC defines the approved set of operations that may be applied to functional requirements: iteration, assignment, selection, and refinement.
    - Iteration: allows a component to be used more than once with varying operations. In the ST, iteration is indicated by a letter in parenthesis placed at the end of the component. For example FDP_ACC.1a and FDP_ACC.1b indicate that the ST includes two iterations of the FDP_ACC.1 requirement, a and b.
    - Assignment: allows the specification of an identified parameter. Assignments are indicated using bold and are surrounded by brackets (e.g., [**assignment**]).
    - Selection: allows the specification of one or more elements from a list. Selections are indicated using bold italics and are surrounded by brackets (e.g., [***selection***]).
    - Refinement: allows the addition of details. Refinements are indicated using bold, for additions, and strike-through, for deletions (e.g., "… **all** objects …" or "… ~~some~~ **big** things …").
- Other sections of the ST – Other sections of the ST use bolding to highlight text of special interest, such as captions.

2. TOE Description

The Target of Evaluation (TOE) is Red Hat Enterprise Linux (RHEL) Advanced Server (AS) Version 3 Update 5 Running on Unisys ES7000 Hardware.

RHEL AS is a general purpose, multi-user, multi-tasking Linux based operating system. In conjunction with its underlying hardware provides a platform for a variety of applications in the governmental and commercial environment. RHEL AS is available on a broad range of computer systems, ranging from departmental servers to multi-processor servers, but this evaluated configuration is based on Unisys ES7000 mainframes.

2.1 TOE Overview

The TOE Security Functions (TSF) consist of functions of RHEL (and its supporting hardware) that run in trusted domains that RHEL, in conjunction with the underlying hardware, keeps separate from potentially untrusted users and their processes. These are the primary functions that enforce the security policy as defined in this Security Target.

Tools and commands executed outside the RHEL kernel and system processes that are used by an administrative user are trusted to manage the system in a secure manner and as such as also considered part of the TSF insofar as they provide the necessary Security Management security functions. System administration tools include the standard command-line commands. No graphical user interface for system administration or any other operation is included in the TOE.

2.2 TOE Architecture

RHEL AS may be viewed as a series of layers. At the lowest layer, the kernel interacts with the hardware platform, providing a common set of services to application programs. These services include managing system memory, sharing access to the system processor(s), and access to devices. In addition, the operating system provides basic services such as:

- File systems organized within a hierarchy of directories;
- Device drivers providing interfaces to hardware devices;
- User interfaces to run programs and access the file system. RHEL AS includes both graphical interfaces (GNOME and KDE) and shell command interpreters (e.g. bash). Note, however, that the graphical interfaces are excluded from the evaluated configuration; and,
- System utilities to mount file systems, configure networks, and scheduled future tasks.

The kernel operates in the hardware processor's privileged mode, known as kernel mode, with full control of all resources of the underlying hardware. The kernel is access via system calls which are facilitated via system libraries that translate widely known UNIX and POSIX calls, for example, appropriately.

RHEL AS also includes a large number of programs that run in user mode (the hardware processor's unprivileged mode). These include system services, administrator and system utilities, and user programs. Some of the system services and utilities may be invoked just once, to initialize and configure some aspect of the system, whereas others (e.g., daemons) may run permanently (e.g. to accept login requests or update log files). User programs are provided to perform everyday tasks such as listing directories, moving files, or more complex operations such as text editing and, while included in the RHEL AS product, are not considered part of the TOE unless they are required to be used by an administrator.

The kernel is implemented as a series of modules that can be loaded and unloaded to run in privileged mode on the hardware processor. These modules include, for example, device drivers.

Memory management services are provided to handle the allocation of physical memory, either as pages, groups of pages or small blocks; and for allocation of virtual memory, mapped into the address space of running processes.

The Virtual Memory System maintains the address space visible to each process. It creates pages of virtual memory on demand, and manages the loading of those pages from disk, and their swapping back out to disk as required

RHEL AS files can be anything capable of handling the input and output of a stream of data. In addition to stored data objects, files include such things as directories, device drivers and network connections. Implementation details of individual file types are managed by the Virtual File System (VFS). VFS provides a consistent view of multiple file systems including ISO-9660 (for CDRom drives), ext3 (for disk drives) and proc (for various kernel constructs, such as processes).

The operating system kernel maintains a single directory hierarchy of files (a file system) for each disk device mounted as a file system, and for each networked file system.

The VFS is also used to store data associated with processes (the proc file system). Each sub-directory corresponds to an active process on the current system, and additional directories and text files are used to store information about the kernel and loaded drivers. This enables programs to access this information directly in an unprivileged mode.

All device drivers in RHEL AS appear as normal files, and are of three types:

a) Block devices – allowing random access to independent, fixed-size blocks of data e.g. for hard disks;

b) Character devices – most other devices that interact by a stream of characters e.g. mouse, tape; and,

c) Network devices – used by the kernel's networking subsystem, and accessed only indirectly by users.

2.2.1 Physical Boundaries

The TOE is a hardware platform running an operating system. As such, it has a physical interface providing direct access to physical devices, controls, and connectors. Note, however, that in the evaluated configuration it is assumed that the TOE will be physically protected from tampering attempts. As a result, the physical boundaries available to users include connected terminals and network connections where logical interfaces offer an opportunity to log into the TOE.

2.2.2 Logical Boundaries

The logical boundaries of the TOE are represented by the available security functions. Users are required to be identified and authenticated prior to having access to any other functions of the TOE. Attempts to authenticate and subsequent use of security functions is audited in a manner that is essentially transparent to the users. Once authenticated, processes representing users are limited by access controls and also by additional restrictions (e.g., the inability to tamper with the kernel or other user processes). If the user is an administrator, the TOE provides a number of interfaces that can be used to manage and control the management of its own security functions.

2.2.2.1 Security Audit

The TOE provides an audit capability to generate audit records for security relevant events. The administrative user can select which events will be audited and for which users auditing is active.

The TOE provides tools for the administrative user that allow him to extract specific types of audit events, audit events for specific users and groups, and successful or failed events from the overall audit records collected by the TOE. Those tools allow an administrative user to save or print the selected audit records in human readable form.

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

2.2.2.2 User Data Protection

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

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

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

2.2.2.3 Identification and Authentication

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

2.2.2.4 Security Management

The management of the security critical parameters of the TOE is performed by administrative users. A set of commands that require root privileges are used for system management. Security parameters are stored in specific files that are protected by the access control mechanisms of the TOE against unauthorized access by users.

2.2.2.5 Protection of the TOE Security Functions

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

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

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

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

2.3 TOE Documentation

Unisys offers a series of documents that describe the installation process for the TOE as well as guidance for subsequent use and administration of the applicable security features. Refer to Section 6 for information about these and other documentation associated with the TOE.

3. Security Environment

The intended environment of the TOE is as defined in the Controlled Access Protection Profile (CAPP) Version 1.d, 8 October 1999 (CAPP). Note that while the EAL 3 augmented with ALC_FLR.2 assurance target does not change the stated policy statements or alleviate the assumptions, it does serve to add confidence in conformance with the policies.

3.1 Organizational Policies

P.ACCOUNTABILITY	The users of the system shall be held accountable for their actions within the system.
P.AUTHORIZED_USERS	Only those users who have been authorized to access the information within the system may access the system.
P.NEED_TO_KNOW	The system must limit the access to, modification of, and destruction of the information in protected resources to those authorized users which have a 'need to know' for that information.

3.2 Assumptions

A.CONNECT	All connections to peripheral devices reside within the controlled access facilities. CAPP-conformant TOEs only address security concerns related to the manipulation of the TOE through its authorized access points. Internal communication paths to access points such as terminals are assumed to be adequately protected.
A.COOP	Authorized users possess the necessary authorization to access at least some of the information managed by the TOE and are expected to act in a cooperating manner in a benign environment.
A.LOCATE	The processing resources of the TOE will be located within controlled access facilities which will prevent unauthorized physical access.
A.MANAGE	There will be one or more competent individuals assigned to manage the TOE and the security of the information it contains.
A.NO_EVIL_ADM	The system administrative personnel are not careless, willfully negligent, or hostile, and will follow and abide by the instructions provided by the administrator documentation.
A.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. CAPP-conformant TOEs are applicable to networked or distributed environments only if the entire network operates under the same constraints and resides within a single management domain. There are no security requirements which address the need to trust external systems or the communications links to such systems.
A.PROTECT	The TOE hardware and software critical to security policy enforcement will be protected from unauthorized physical modification.

4. Security Objectives

The following objectives for the TOE and its environment have all been drawn from the Controlled Access Protection Profile (CAPP) Version 1.d, 8 October 1999 (CAPP).

4.1 Security Objectives for the TOE

O.AUDITING	The TSF must record the security relevant actions of users of the TOE. The TSF must present this information to authorized administrators.
O.AUTHORIZATION	The TSF must ensure that only authorized users gain access to the TOE and its resources.
O.DISCRETIONARY_ACCESS	The TSF must control access to resources based on identity of users. The TSF must allow authorized users to specify which resources may be accessed by which users.
O.ENFORCEMENT	The TSF must be designed and implemented in a manner which ensures that the organizational policies are enforced in the target environment.
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.
O.RESIDUAL_INFORMATION	The TSF must ensure that any information contained in a protected resource is not released when the resource is recycled.

4.2 Security Objectives for the Environment

O.CREDEN	Those responsible for the TOE must ensure that all access credentials, such as passwords or other authentication information, are protected by the users in a manner which maintains IT security objectives.
O.INSTALL	Those responsible for the TOE must ensure that the TOE is delivered, installed, managed, and operated in a manner which maintains IT security objectives.
O.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.

5. IT Security Requirements

The following requirements have been drawn from the Controlled Access Protection Profile (CAPP) Version 1.d, 8 October 1999 (CAPP) with the exception of FIA_UAU.2 and FIA_UID.2. FIA_UAU.2 and FIA_UID.2 are drawn from the Common Criteria and serve to replace their weaker counterparts required in the CAPP. In each requirement, all operations left incomplete in the CAPP have been completed.

5.1 TOE Security Functional Requirements

Requirement Class	Requirement Component
FAU: Security Audit	FAU_GEN.1: Audit Data Generation
	FAU_GEN.2: User Identity Association
	FAU_SAR.1: Audit Review
	FAU_SAR.2: Restricted Audit Review
	FAU_SAR.3: Selectable Audit Review
	FAU_SEL.1: Selective Audit
	FAU_STG.1: Guarantees of Audit Data Availability
	FAU_STG.3: Action in Case of Possible Audit Data Loss
	FAU_STG.4: Prevention of Audit Data Loss
FDP: User Data Protection	FDP_ACC.1: Discretionary Access Control Policy
	FDP_ACF.1: Discretionary Access Control Functions
	FDP_RIP.2a: Object Residual Information Protection
	FDP_RIP.2b: Subject Residual Information Protection
FIA: Identification and Authentication	FIA_ATD.1: User Attribute Definition
	FIA_SOS.1: Strength of Authentication Data
	FIA_UAU.2: User authentication before any action
	FIA_UAU.7: Protected Authentication Feedback
	FIA_UID.2: User identification before any action
	FIA_USB.1: User-Subject Binding
FMT: Security Management	FMT_MSA.1: Management of Object Security Attributes
	FMT_MSA.3: Static Attribute Initialization
	FMT_MTD.1a: Management of the Audit Trail
	FMT_MTD.1b: Management of Audited Events
	FMT_MTD.1c: Management of User Attributes
	FMT_MTD.1d: Management of Authentication Data
	FMT_REV.1a: Revocation of User Attributes
	FMT_REV.1b: Revocation of Object Attributes
	FMT_SMR.1: Security Management Roles
FPT: Protection of the TOE Security Functions	FPT_AMT.1: Abstract Machine Testing
	FPT_RVM.1: Reference Mediation
	FPT_SEP.1: Domain Separation
	FPT_STM.1: Reliable Time Stamps

Table 1 TOE Security Functional Components

5.1.1 Security Audit (FAU)

5.1.1.1 Audit Data Generation (FAU_GEN.1)

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

FAU_GEN.1.2 The TSF shall record within each audit record at least the following information:[**a) Date and time of the event, type of event, subject identity, and the outcome (success or failure) of the event; b) The additional information specified in the 'Details' column of Table 1 (Auditable Events) of the Controlled Access Protection Profile**].

5.1.1.2 User Identity Association (FAU_GEN.2)

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

5.1.1.3 Audit Review (FAU_SAR.1)

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

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

5.1.1.4 Restricted Audit Review (FAU_SAR.2)

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

5.1.1.5 Selectable Audit Review (FAU_SAR.3)

FAU_SAR.3.1 The TSF shall provide the ability to perform [***searches***] of audit data based on the following attributes: a) User identity; b) [**group identifier (real and effective); c) event type; and, d) outcome (success/failure)**].

5.1.1.6 Selective Audit (FAU_SEL.1)

FAU_SEL.1.1 The TSF shall be able to include or exclude auditable events from the set of audited events based on the following attributes:[**a) User identity; b) system call; and, c) file name**].

5.1.1.7 Guarantees of Audit Data Availability (FAU_STG.1)

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

FAU_STG.1.2 The TSF shall be able to[***prevent***] unauthorised modifications to the audit records in the audit trail. *(per International Interpretations #141 and #202)*

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

FAU_STG.3.1 The TSF shall ~~take~~ [**generate an alarm to the authorized administrator if the audit trail exceeds an administrator defined value for the minimum space required for the file system containing the audit log file**].

5.1.1.9 Prevention of Audit Data Loss (FAU_STG.4)

FAU_STG.4.1 The TSF shall **be able to** [***prevent auditable events, except those taken by the authorized administrator***], and [**stop each process as it attempts to generate an audit record**] if the audit trail is full.

5.1.2 User Data Protection (FDP)

5.1.2.1 Discretionary Access Control Policy (FDP_ACC.1)

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

5.1.2.2 Discretionary Access Control Functions (FDP_ACF.1)

FDP_ACF.1.1 The TSF shall enforce the [**Discretionary Access Control Policy**] to objects based on the following:[**a) The user identity and group membership(s) associated with a subject; and b) The following access control attributes associated with an object: File system objects: POSIX ACLs and permission bits (ACLs can be used to grant or deny access to the granularity of a single user or group using Access Control Entries. Those ACL entries include the standard Unix permission bits. Posix ACLs can be used for file system objects within the ext3 file system) - Access rights for file system objects are: read, write, execute (ordinary files), and search (directories); IPC objects: permission bits - Access rights for IPC objects are: read and write**].

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

File system objects within the local file system

1) A subject must have search permission for every element of the pathname and the requested access for the object.

2) A subject has a specific type of access to an object if:

a) The subject has been granted access according to the ACL_USER_OBJ or ACL_OTHER type entry in the ACL of the object OR

b) The subject has been granted access by an ACL_USER, ACL_GROUP_OBJ or ACL_GROUP entry and the associated right is also granted by the ACL_MASK entry of the ACL if the ACL_MASK entry exist OR 3) The subject has been granted access by the ACL_GROUP_OBJ entry and no ACL_MASK entry exists in the ACL of the object;

File system objects in other file systems

1) A subject must have search permission for every element of the pathname and the requested access for the object.

2) A subject has a specific type access to an object if:

a) The subject has the effective userid of the owner of the object and the requested type of access is within the permission bits defined for the owner OR

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

c) The subject has neither the effective userid of the owner of the object nor is the effective group id identical to the file system object group id and requested type of access is within the permission bits defined for 'world'; IPC objects: Access permissions are defined by permission bits of the IPC object. The process creating the object defines the creator, owner and group based on the userid of the current process.

Process access to an IPC object

1) Access is permitted if the effective userid of the of the current process is equal to the userid of the IPC object creator or owner and the 'owner' permission bit for the requested type of access is set OR

2) The effective userid of the current process is not equal to the userid of the IPC object creator or owner and the effective group id of the current process is equal to the group id of the IPC object and the 'group' permission bit for the requested type of access is set OR

3) The 'world' permission bit for the requested type of access is set for users that do not satisfy one of the first two conditions.]

FDP_ACF.1.3 The TSF shall explicitly authorize access of subjects to objects based in the following additional rules: [**If the user is root, the requested file system or IPC object access is allowed. If execution permission has been granted for a file system object, the requested access is allowed**.].

FDP_ACF.1.4 The TSF shall explicitly deny access of subjects to objects based on the [**following rules: Write access to file system objects on a file system mounted as read-only is always denied AND Write access to a file marked as immutable is always denied**.].

5.1.2.3 Object Residual Information Protection (FDP_RIP.2a)

FDP_RIP.2a.1 The TSF shall ensure that any previous information content of a resource is made unavailable upon the [***allocation of the resource to***] all objects.

5.1.2.4 Subject Residual Information Protection (FDP_RIP.2b)

FDP_RIP.2b.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.

5.1.3 Identification and Authentication (FIA)

5.1.3.1 User Attribute Definition (FIA_ATD.1)

FIA_ATD.1.1 The TSF shall maintain the following list of security attributes belonging to individual users:[**a) User Identifier; b) Group Memberships; c) Authentication Data; d) Security-relevant Roles**].

5.1.3.2 Strength of Authentication Data (FIA_SOS.1)

FIA_SOS.1.1 The TSF shall provide a mechanism to verify that secrets meet [**the following: a) For each attempt to use the authentication mechanism, the probability that a random attempt will succeed is less than one in 1,000,000; b) For multiple attempts to use the authentication mechanism during a one minute period, the probability that a random attempt during that minute will succeed is less than one in 100,000; and c) Any feedback given during an attempt to use the authentication mechanism will not reduce the probability below the above metrics**].

5.1.3.3 User authentication before any action (FIA_UAU.2)

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

5.1.3.4 Protected Authentication Feedback (FIA_UAU.7)

FIA_UAU.7.1 The TSF shall provide only [**obscured feedback**] to the user while the authentication is in progress.

5.1.3.5 User identification before any action (FIA_UID.2)

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

5.1.3.6 User-Subject Binding (FIA_USB.1)

FIA_USB.1.1 The TSF shall associate the following user security attributes with subjects acting on the behalf of that user: [**no other security attributes**]. *(per International Interpretation #137)*

FIA_USB.1.2 The TSF shall enforce the following rules on the initial association of user security attributes with subjects acting on the behalf of users: [**a) Upon successful identification and authentication, the login user ID, the real user ID and the effective user ID shall be those specified in the user entry for the user that has authenticated successfully; and b) Upon successful identification and authentication, the real group ID and the effective group ID shall be those specified via the group membership attribute in the user entry**]. *(per International Interpretation #137)*

FIA_USB.1.3 The TSF shall enforce the following rules governing changes to the user security attributes associated with subjects acting on the behalf of users: [**a) The effective user ID of a user can be changed by the use of an executable with the setuid bit set. In this case the program is executed with the effective user ID of the program owner. Access rights are then evaluated using the effective user ID of the program owner. The real and login user ID remain unchanged; b) The effective and real user ID of a user can be changed by the su command. In this case the real and effective user ID of the user is changed to the user specified in the su command (provided authentication is successful). The login user ID remains unchanged; and, c) The effective group ID of a user can be changed by the use of an executable with the setgid bit set. In this case the program is executed with the effective group ID of the program owner. Access rights are then evaluated using the effective group ID of the program owner**]. *(per International Interpretation #137)*

5.1.4 Security Management (FMT)

5.1.4.1 Management of Object Security Attributes (FMT_MSA.1)

FMT_MSA.1.1 The TSF shall enforce the [**Discretionary Access Control Policy**] to restrict the ability to [***modify***] the ~~security~~ **access control attributes associated with a named object** to [**administrative users, the owner of the object, and for IPC objects the original creator of the object**].

5.1.4.2 Static Attribute Initialization (FMT_MSA.3)

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

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

5.1.4.3 Management of the Audit Trail (FMT_MTD.1a)

FMT_MTD.1a.1 The TSF shall restrict the ability to [***create, delete, and clear]*** the [**audit trail**] to [**authorized administrators**].

5.1.4.4 Management of Audited Events (FMT_MTD.1b)

FMT_MTD.1b.1 The TSF shall restrict the ability to [***modify or [*observe**]] the [**set of audited events**] to [**authorized administrators**].

5.1.4.5 Management of User Attributes (FMT_MTD.1c)

FMT_MTD.1c.1 The TSF shall restrict the ability to [[**initialize] *and modify***] the [**user security attributes, other than authentication data**], to[**authorized administrators**].

5.1.4.6 Management of Authentication Data (FMT_MTD.1d)

FMT_MTD.1d.1 The TSF shall restrict the ability to [**initialize**] the [**authentication data**] to [**authorized administrators**].

FMT_MTD.1d.2 The TSF shall restrict the ability to [***modify***] the [**authentication data**] to[**the following: a) authorized administrators; and b) users authorized to modify their own authentication data**].

5.1.4.7 Revocation of User Attributes (FMT_REV.1a)

FMT_REV.1a.1 The TSF shall restrict the ability to revoke security attributes associated with the [***users***] within the TSC to [**authorized administrators**].

FMT_REV.1a.2 The TSF shall enforce the rules: [**a) The immediate revocation of security-relevant authorizations; and b) Revocations/modifications made by an authorized administrator to security attributes of a user like the user identifier, user name, user group(s), user password or user login shell shall be effective the next time the user logs in**].

5.1.4.8 Revocation of Object Attributes (FMT_REV.1b)

FMT_REV.1b.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.1b.2 The TSF shall enforce the rules: [**a) The access rights associated with an object shall be enforced when an access check is made; and b) Access rights to file system and IPC objects are checked when the object is opened. Revocations of access rights for file system objects become effective the next time a user affected by the revocation tries to open a file system object**].

5.1.4.9 Security Management Roles (FMT_SMR.1)

FMT_SMR.1.1 The TSF shall maintain the roles:[**a) authorized administrator; b) users authorized by the Discretionary Access Control Policy to modify object security attributes; c) users authorized to modify their own authentication data**].

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

5.1.5 Protection of the TOE Security Functions (FPT)

5.1.5.1 Abstract Machine Testing (FPT_AMT.1)

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

5.1.5.2 Reference Mediation (FPT_RVM.1)

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

5.1.5.3 Domain Separation (FPT_SEP.1)

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

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

5.1.5.4 Reliable Time Stamps (FPT_STM.1)

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

5.2 TOE Security Assurance Requirements

The security assurance requirements for the TOE are the EAL 3 augmented with ALC_FLR.2 components as specified in Part 3 of the Common Criteria. No operations are applied to the assurance components.

Requirement Class	Requirement Component
ACM: Configuration management	ACM_CAP.3: Authorisation controls
	ACM_SCP.1: TOE CM coverage
ADO: Delivery and operation	ADO_DEL.1: Delivery procedures
	ADO_IGS.1: Installation, generation, and start-up procedures
ADV: Development	ADV_FSP.1: Informal functional specification
	ADV_HLD.2: Security enforcing high-level design
	ADV_RCR.1: Informal correspondence demonstration
AGD: Guidance documents	AGD_ADM.1: Administrator guidance
	AGD_USR.1: User guidance
ALC: Life cycle support	ALC_DVS.1: Identification of security measures
	ALC_FLR.2: Flaw reporting procedures
ATE: Tests	ATE_COV.2: Analysis of coverage
	ATE_DPT.1: Testing: high-level design
	ATE_FUN.1: Functional testing
	ATE_IND.2: Independent testing - sample
AVA: Vulnerability assessment	AVA_MSU.1: Examination of guidance
	AVA_SOF.1: Strength of TOE security function evaluation
	AVA_VLA.1: Developer vulnerability analysis

Table 2 EAL 3 augmented with ALC_FLR.2 Assurance Components

5.2.1 Configuration management (ACM)

5.2.1.1 Authorization controls (ACM_CAP.3)

ACM_CAP.3.1d The developer shall provide a reference for the TOE.
ACM_CAP.3.2d The developer shall use a CM system.
ACM_CAP.3.3d The developer shall provide CM documentation.
ACM_CAP.3.1c The reference for the TOE shall be unique to each version of the TOE.
ACM_CAP.3.2c The TOE shall be labelled with its reference.
ACM_CAP.3.3c The CM documentation shall include a configuration list and a CM plan.
ACM_CAP.3.4c The configuration list shall uniquely identify all configuration items that comprise the TOE.
ACM_CAP.3.5c The configuration list shall describe the configuration items that comprise the TOE.
ACM_CAP.3.6c The CM documentation shall describe the method used to uniquely identify the configuration items.
ACM_CAP.3.7c The CM system shall uniquely identify all configuration items.
ACM_CAP.3.8c The CM plan shall describe how the CM system is used.
ACM_CAP.3.9c The evidence shall demonstrate that the CM system is operating in accordance with the CM plan.

ACM_CAP.3.10c The CM documentation shall provide evidence that all configuration items have been and are being effectively maintained under the CM system.
ACM_CAP.3.11c The CM system shall provide measures such that only authorised changes are made to the configuration items.
ACM_CAP.3.1e The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.1.2 TOE CM coverage (ACM_SCP.1)

ACM_SCP.1.1d The developer shall provide a list of configuration items for the TOE.
ACM_SCP.1.1c The list of configuration items shall include the following: implementation representation and the evaluation evidence required by the assurance components in the ST.
ACM_SCP.1.1e The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.2 Delivery and operation (ADO)

5.2.2.1 Delivery procedures (ADO_DEL.1)

ADO_DEL.1.1d The developer shall document procedures for delivery of the TOE or parts of it to the user.
ADO_DEL.1.2d The developer shall use the delivery procedures.
ADO_DEL.1.1c The delivery documentation shall describe all procedures that are necessary to maintain security when distributing versions of the TOE to a user's site.
ADO_DEL.1.1e The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.2.2 Installation, generation, and start-up procedures (ADO_IGS.1)

ADO_IGS.1.1d The developer shall document procedures necessary for the secure installation, generation, and start-up of the TOE.
ADO_IGS.1.1c The installation, generation and start-up documentation shall describe all the steps necessary for secure installation, generation and start-up of the TOE.
ADO_IGS.1.1e The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
ADO_IGS.1.2e The evaluator shall determine that the installation, generation, and start-up procedures result in a secure configuration.

5.2.3 Development (ADV)

5.2.3.1 Informal functional specification (ADV_FSP.1)

ADV_FSP.1.1d The developer shall provide a functional specification.
ADV_FSP.1.1c The functional specification shall describe the TSF and its external interfaces using an informal style.
ADV_FSP.1.2c The functional specification shall be internally consistent.
ADV_FSP.1.3c The functional specification shall describe the purpose and method of use of all external TSF interfaces, providing details of effects, exceptions and error messages, as appropriate.
ADV_FSP.1.4c The functional specification shall completely represent the TSF.
ADV_FSP.1.1e The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
ADV_FSP.1.2e The evaluator shall determine that the functional specification is an accurate and complete instantiation of the TOE security functional requirements.

5.2.3.2 Security enforcing high-level design (ADV_HLD.2)

ADV_HLD.2.1d The developer shall provide the high-level design of the TSF.
ADV_HLD.2.1c The presentation of the high-level design shall be informal.
ADV_HLD.2.2c The high-level design shall be internally consistent.

ADV_HLD.2.3c The high-level design shall describe the structure of the TSF in terms of subsystems.
ADV_HLD.2.4c The high-level design shall describe the security functionality provided by each subsystem of the TSF.
ADV_HLD.2.5c The high-level design shall identify any underlying hardware, firmware, and/or software required by the TSF with a presentation of the functions provided by the supporting protection mechanisms implemented in that hardware, firmware, or software.
ADV_HLD.2.6c The high-level design shall identify all interfaces to the subsystems of the TSF.
ADV_HLD.2.7c The high-level design shall identify which of the interfaces to the subsystems of the TSF are externally visible.
ADV_HLD.2.8c The high-level design shall describe the purpose and method of use of all interfaces to the subsystems of the TSF, providing details of effects, exceptions and error messages, as appropriate.
ADV_HLD.2.9c The high-level design shall describe the separation of the TOE into TSP-enforcing and other subsystems.
ADV_HLD.2.1e The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
ADV_HLD.2.2e The evaluator shall determine that the high-level design is an accurate and complete instantiation of the TOE security functional requirements.

5.2.3.3 Informal correspondence demonstration (ADV_RCR.1)

ADV_RCR.1.1d The developer shall provide an analysis of correspondence between all adjacent pairs of TSF representations that are provided.
ADV_RCR.1.1c For each adjacent pair of provided TSF representations, the analysis shall demonstrate that all relevant security functionality of the more abstract TSF representation is correctly and completely refined in the less abstract TSF representation.
ADV_RCR.1.1e The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.4 Guidance documents (AGD)

5.2.4.1 Administrator guidance (AGD_ADM.1)

AGD_ADM.1.1d The developer shall provide administrator guidance addressed to system administrative personnel.
AGD_ADM.1.1c The administrator guidance shall describe the administrative functions and interfaces available to the administrator of the TOE.
AGD_ADM.1.2c The administrator guidance shall describe how to administer the TOE in a secure manner.
AGD_ADM.1.3c The administrator guidance shall contain warnings about functions and privileges that should be controlled in a secure processing environment.
AGD_ADM.1.4c The administrator guidance shall describe all assumptions regarding user behaviour that are relevant to secure operation of the TOE.
AGD_ADM.1.5c The administrator guidance shall describe all security parameters under the control of the administrator, indicating secure values as appropriate.
AGD_ADM.1.6c The administrator guidance shall describe each type of security-relevant event relative to the administrative functions that need to be performed, including changing the security characteristics of entities under the control of the TSF.
AGD_ADM.1.7c The administrator guidance shall be consistent with all other documentation supplied for evaluation.
AGD_ADM.1.8c The administrator guidance shall describe all security requirements for the IT environment that are relevant to the administrator.
AGD_ADM.1.1e The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.4.2 User guidance (AGD_USR.1)

AGD_USR.1.1d The developer shall provide user guidance.
AGD_USR.1.1c The user guidance shall describe the functions and interfaces available to the non-administrative users of the TOE.

AGD_USR.1.2c The user guidance shall describe the use of user-accessible security functions provided by the TOE.
AGD_USR.1.3c The user guidance shall contain warnings about user-accessible functions and privileges that should be controlled in a secure processing environment.
AGD_USR.1.4c The user guidance shall clearly present all user responsibilities necessary for secure operation of the TOE, including those related to assumptions regarding user behaviour found in the statement of TOE security environment.
AGD_USR.1.5c The user guidance shall be consistent with all other documentation supplied for evaluation.
AGD_USR.1.6c The user guidance shall describe all security requirements for the IT environment that are relevant to the user.
AGD_USR.1.1e The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.5 Life cycle support (ALC)

5.2.5.1 Identification of security measures (ALC_DVS.1)

ALC_DVS.1.1d The developer shall produce development security documentation.
ALC_DVS.1.1c The development security documentation shall describe all the physical, procedural, personnel, and other security measures that are necessary to protect the confidentiality and integrity of the TOE design and implementation in its development environment.
ALC_DVS.1.2c The development security documentation shall provide evidence that these security measures are followed during the development and maintenance of the TOE.
ALC_DVS.1.1e The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
ALC_DVS.1.2e The evaluator shall confirm that the security measures are being applied.

5.2.5.2 Flaw reporting procedures (ALC_FLR.2)

ALC_FLR.2.1d The developer shall provide flaw remediation procedures addressed to TOE developers.
ALC_FLR.2.2d The developer shall establish a procedure for accepting and acting upon all reports of security flaws and requests for corrections to those flaws.
ALC_FLR.2.3d The developer shall provide flaw remediation guidance addressed to TOE users.
ALC_FLR.2.1c The flaw remediation procedures documentation shall describe the procedures used to track all reported security flaws in each release of the TOE.
ALC_FLR.2.2c The flaw remediation procedures shall require that a description of the nature and effect of each security flaw be provided, as well as the status of finding a correction to that flaw.
ALC_FLR.2.3c The flaw remediation procedures shall require that corrective actions be identified for each of the security flaws.
ALC_FLR.2.4c The flaw remediation procedures documentation shall describe the methods used to provide flaw information, corrections and guidance on corrective actions to TOE users.
ALC_FLR.2.5c The flaw remediation procedures documentation shall describe a means by which the developer receives from TOE users reports and enquiries of suspected security flaws in the TOE.
ALC_FLR.2.6c The procedures for processing reported security flaws shall ensure that any reported flaws are corrected and the correction issued to TOE users.
ALC_FLR.2.7c The procedures for processing reported security flaws shall provide safeguards that any corrections to these security flaws do not introduce any new flaws.
ALC_FLR.2.8c The flaw remediation guidance shall describe a means by which TOE users report to the developer any suspected security flaws in the TOE.
ALC_FLR.2.1e The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.6 Tests (ATE)

5.2.6.1 Analysis of coverage (ATE_COV.2)

ATE_COV.2.1d The developer shall provide an analysis of the test coverage.
ATE_COV.2.1c The analysis of the test coverage shall demonstrate the correspondence between the tests identified in the test documentation and the TSF as described in the functional specification.
ATE_COV.2.2c The analysis of the test coverage shall demonstrate that the correspondence between the TSF as described in the functional specification and the tests identified in the test documentation is complete.
ATE_COV.2.1e The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.6.2 Testing: high-level design (ATE_DPT.1)

ATE_DPT.1.1d The developer shall provide the analysis of the depth of testing.
ATE_DPT.1.1c The depth analysis shall demonstrate that the tests identified in the test documentation are sufficient to demonstrate that the TSF operates in accordance with its high-level design.
ATE_DPT.1.1e The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.6.3 Functional testing (ATE_FUN.1)

ATE_FUN.1.1d The developer shall test the TSF and document the results.
ATE_FUN.1.2d The developer shall provide test documentation.
ATE_FUN.1.1c The test documentation shall consist of test plans, test procedure descriptions, expected test results and actual test results.
ATE_FUN.1.2c The test plans shall identify the security functions to be tested and describe the goal of the tests to be performed.
ATE_FUN.1.3c The test procedure descriptions shall identify the tests to be performed and describe the scenarios for testing each security function. These scenarios shall include any ordering dependencies on the results of other tests.
ATE_FUN.1.4c The expected test results shall show the anticipated outputs from a successful execution of the tests.
ATE_FUN.1.5c The test results from the developer execution of the tests shall demonstrate that each tested security function behaved as specified.
ATE_FUN.1.1e The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.6.4 Independent testing - sample (ATE_IND.2)

ATE_IND.2.1d The developer shall provide the TOE for testing.
ATE_IND.2.1c The TOE shall be suitable for testing.
ATE_IND.2.2c The developer shall provide an equivalent set of resources to those that were used in the developer's functional testing of the TSF.
ATE_IND.2.1e The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
ATE_IND.2.2e The evaluator shall test a subset of the TSF as appropriate to confirm that the TOE operates as specified.
ATE_IND.2.3e The evaluator shall execute a sample of tests in the test documentation to verify the developer test results.

5.2.7 Vulnerability assessment (AVA)

5.2.7.1 Examination of guidance (AVA_MSU.1)

AVA_MSU.1.1d The developer shall provide guidance documentation.

AVA_MSU.1.1c The guidance documentation shall identify all possible modes of operation of the TOE (including operation following failure or operational error), their consequences and implications for maintaining secure operation.
AVA_MSU.1.2c The guidance documentation shall be complete, clear, consistent and reasonable.
AVA_MSU.1.3c The guidance documentation shall list all assumptions about the intended environment.
AVA_MSU.1.4c The guidance documentation shall list all requirements for external security measures (including external procedural, physical and personnel controls).
AVA_MSU.1.1e The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
AVA_MSU.1.2e The evaluator shall repeat all configuration and installation procedures to confirm that the TOE can be configured and used securely using only the supplied guidance documentation.
AVA_MSU.1.3e The evaluator shall determine that the use of the guidance documentation allows all insecure states to be detected.

5.2.7.2 Strength of TOE security function evaluation (AVA_SOF.1)

AVA_SOF.1.1d The developer shall perform a strength of TOE security function analysis for each mechanism identified in the ST as having a strength of TOE security function claim.
AVA_SOF.1.1c For each mechanism with a strength of TOE security function claim the strength of TOE security function analysis shall show that it meets or exceeds the minimum strength level defined in the PP/ST.
AVA_SOF.1.2c For each mechanism with a specific strength of TOE security function claim the strength of TOE security function analysis shall show that it meets or exceeds the specific strength of function metric defined in the PP/ST.
AVA_SOF.1.1e The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
AVA_SOF.1.2e The evaluator shall confirm that the strength claims are correct.

5.2.7.3 Developer vulnerability analysis (AVA_VLA.1)

AVA_VLA.1.1d The developer shall perform a vulnerability analysis.
AVA_VLA.1.2d The developer shall provide vulnerability analysis documentation.
AVA_VLA.1.1c The vulnerability analysis documentation shall describe the analysis of the TOE deliverables performed to search for obvious ways in which a user can violate the TSP.
AVA_VLA.1.2c The vulnerability analysis documentation shall describe the disposition of obvious vulnerabilities.
AVA_VLA.1.3c The vulnerability analysis documentation shall show, for all identified vulnerabilities, that the vulnerability cannot be exploited in the intended environment for the TOE.
AVA_VLA.1.1e The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
AVA_VLA.1.2e The evaluator shall conduct penetration testing, building on the developer vulnerability analysis, to ensure obvious vulnerabilities have been addressed.

6. TOE Summary Specification

This chapter describes the security functions and associated assurance measures.

6.1 TOE Security Functions

6.1.1 Security Audit

The TOE audit function is implemented using *systrace*. Systrace is a Linux security feature that works by interactively restricting process functionality based on administrator configured policies. Systrace does this by limiting one or more processes' access to the system and kernel functions, and alerts the system of any attempts to violate the policy. Initially, the administrator must carefully define the systrace policies so as to provide the necessary functionality to the process without compromising the security of the system.

The TOE allows configuring the events to be actually audited from the set of auditable events using rules defined in an audit configuration file using predicates and logical operations. This allows for a very flexible definition of the events to be audited and the conditions under which events are audited (e.g., for specific system calls or files). The system administrator is also able to define a set of user Ids for which auditing is active or inactive. The configuration file is restricted so that it is accessible only by an administrative user. In addition trusted processes can generate audit records and send them to the kernel to be placed in the audit trail.

Auditing is performed on a per process basis. A process can enable or disabling auditing for itself by attaching itself or detaching itself to the audit subsystem provided it is running with root privileges and the attribute of being attached to the audit subsystem is inherited by all child processes, which ensures that an unprivileged process cannot alter its state of being audited. The login ID is associated with each audit events ensuring that events can be easily associated with the user authenticated to the TOE, regardless of their current, effective identity.

Each event to be audited is forwarded by the kernel to an audit daemon, which writes the audit records to the audit trail. When the audit queue does not have sufficient space to hold an audit record the responsible process is suspended indefinitely until the queue has enough space again. This ensures that audit records do not get lost due to a resource shortage.

Several fixed length audit files are maintained with a pointer to the current location. The audit records are written until the current file has reached it maximum administrator user configured capacity and then the next file is utilized until the last file reaches its maximum capacity at which point the first file is used again. Whenever such a switch between two files occurs the audit subsystems starts a program defined in the audit configuration file to process the data collected in the audit file (in the evaluated configuration this program is the *aucat* program that converts the binary data into a human readable format and appends the result to an existing audit data file). Note that a *syslog* message will be generated should the file system containing the audit files exceed an administrative user defined threshold.

Auditable events include:

- Start and stop of audit processing
- Authentication attempts and rejection of an unacceptable password
- Use of specific system calls, including
- Attempts to open objects
- Attempts to fork processes
- Attempts to change access and other security attributes
- Attempts to change the time
- Attempts to use calls requiring root privilege
- Exceeding a disk capability threshold or exhausting the available space on a disk
- Administrator operations – events are generated by the administrator utilities indicating use of abstract machine tests and changes to users and their rights and roles, including details of the changed values.

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

Audit ID: unique 32 bit identifier
Login ID: User ID of the user authenticated by the system (regardless if the user has changed his real and/or effective user ID afterwards)
Effective user ID: the effective user ID of the process at the time the audit event was generated
Timestamp: Date and time the audit record was generated
Event Type: The event specific data will always contain data indicating if the request that caused the event has been successful or not.

The Security Audit function is designed to satisfy the following security functional requirements:

- FAU_GEN.1: The required audit events are auditable with the required audit record content.
- FAU_GEN.2: The identity of the responsible user is included in each audit record.
- FAU_SAR.1: The audit records are available in human readable form.
- FAU_SAR.2: The audit records are restricted so that only an administrative user can access them.
- FAU_SAR.3: The TOE provides tools to search the audit records based on user identities, event type and outcome.
- FAU_SEL.1: The TOE allows an administrative user to select which users, system calls, and objects will be audited.
- FAU_STG.1: The audit records are protected so that only an administrative user can access them.
- FAU_STG.3: The TOE allows an administrative user to define a disk capacity threshold which, if exceeded, will cause a *syslog* message to be generated.
- FAU_STG.4: When there is no space to write audit records, each process will suspend as it attempts to generate an audit records, thereby preventing the auditable event from completing until space becomes available.

6.1.2 User Data Protection

The general policy is that processes are allowed only the accesses specified by the class-specific policies. The ability to propagate access permissions is limited to those processes that have those permissions already. However, an administrative user is exempt from all restrictions and can perform any action desired.

Discretionary Access Control (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 RHEL AS. DAC is implemented with permission bits and, when specified, ACLs, and all operations that a process may perform on the following objects are controlled using DAC: file system objects (ordinary files, directories, symbolic links, device special files, UNIX Domain socket special files, named pipes) and IPC objects (message queues, semaphores, shared memory segments).

A process whose effective user ID matches the file owner ID (or creating user ID in the case of IPC objects only) can change the file attributes, permissions, and group (except for read-only file systems, of course). However, a 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, administrative users can make any desired changes to the file attributes, permissions, group, and owning user of the file.

Permission bits are the standard UNIX DAC mechanism and are used on all RHEL 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 within an ext3 file system and provides a finer level of granularity than do permission bits.

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

Permission Bits

RHEL AS supports standard UNIX permission bits to provide one form of DAC for file system objects in the /proc and ISO9660 file systems. There are three sets of three bits that define access for three categories of users: the owning user, users in the owning group, and other users. The three bits in each set indicate the access permissions granted to each user category: one bit for read (r), one for write (w) and one for execute (x). Note that write access to file systems mounted as read only (e. g. CD-ROM) is always rejected. Note also that IPC objects do not have execute access.

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

- Administrative user processes are able to read and write all files, ignoring the permission bits. Administrative user processes are also able to execute any file if it is executable for someone.
- If the effective user ID = object's owning user ID (or creating user ID in the case of IPC objects only) and the owning user permission bits allow the type of access requested access is granted or denied with no further checks.
- If the effective group ID, or any supplementary groups of the process = object's owning group ID, and the owning group permission bits allow the type of access requested access is granted or denied with no further checks.
- If the process is neither the owner nor a member of an appropriate group and the permission bits for world allow the type of access requested, then the subject is permitted access.
- If none of the conditions above are satisfied, and the process is not the root identity, then the access attempt is denied.

Access Control Lists

RHEL AS provides support for POSIX type ACLs for the ext3 file system allowing fine grained access control on a user basis. The semantics of those ACLs is summarized in this section. Note, however, that IPC objects cannot have ACLs, so only permission bits are used to control access to IPC objects.

An ACL entry contains the following information:

1. A tag type that specifies the type of the ACL entry
    - ACL_GROUP: an ACL entry of this type defines access rights for processes whose effective group ID or any supplementary group IDs match the one in the ACL entry qualifier
    - ACL_GROUP_OBJ: an ACL entry of this type defines access rights for processes whose effective group ID or any supplementary group IDs match the group ID of the group of the file
    - ACL_MASK: an ACL entry of this type defines the maximum discretionary access rights a process in the file group class
    - ACL_OTHER: an ACL entry of this type defines access rights for processes whose attributes do not match any other entry in the ACL
    - ACL_USER: an ACL entry of this type defines access rights for processes whose effective user ID matches the ACL entry qualifier
    - ACL_USER_OBJ: an ACL entry of this type defines access rights for processes whose effective user ID matches the user ID of the owner of the file
2. A qualifier that specifies an instance of an ACL entry type
    - Required for ACL entries of type ACL_GROUP and ACL_USER and contain either the user ID or the group ID for which the access rights defined in the entry shall apply.
3. A permission set that specifies the discretionary access rights for processes identified by the tag type and qualifier.

An ACL contains exactly one entry for each of the ACL_USER_OBJ, ACL_GROUP_OBJ, and ACL_OTHER tag type (called the 'required ACL entries'). An ACL may have between zero and a defined maximum number of entries of the type ACL_GROUP and ACL_USER. An ACL that has only the three required ACL entries is called a 'minimum ACL'. ACLs with one or more ACL entries of type ACL_GROUP or ACL_USER are called an 'extended ACL'.

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

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

A default ACL is an additional ACL which may be associated with a directory. This default ACL has no effect on the access to this directory. Instead the default ACL is used to initialize the ACL for any file that is created in this directory. If the new file created is a directory it inherits the default ACL from its parent directory. When an object is created within a directory and the ACL is not defined with the function creating the object, the new object inherits the default ACL of its parent directory as its initial ACL.

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, otherwise the permission bits are used:

1. Write access to a file on a read-only file system will always be denied.
2. Write access to a file with the immutable attribute will always be denied.
3. **If** the effective user ID of the process matches the user ID of the file object owner, **then**
   if the ACL_USER_OBJ entry contains the requested permissions, access is granted,
   else access is denied.
   else if the effective user ID of the process matches the qualifier of any entry of type ACL_USER, **then**
   if the matching ACL_USER entry and the ACL_MASK entry contain the requested permissions, access is granted,
   else access is denied.
   else if the effective group ID or any of the supplementary group IDs of the process match the qualifier of the entry of type ACL_GROUP_OBJ, or the qualifier of any entry of type ACL_GROUP, **then**
   if the ACL_MASK entry and any of the matching ACL_GROUP_OBJ or ACL_GROUP entries contain all the requested permissions, access is granted,
   else access is denied.
   else if the ACL_OTHER entry contains the requested permissions, access is granted.
   else access is denied**.**

Access Revocation

File system objects access checks are performed when the object is initially opened, and are not checked on each subsequent access. Changes to access controls (i.e., revocation) are effective with the next attempt to open the object. In cases where an administrative user determines that immediate revocation of access to a file system object is required, the administrative user can reboot the computer, resulting in a close on the object and forcing an open of the object on system reboot.

Object Reuse
RHEL AS uses an explicit initialization scheme for most TSF-managed abstractions, where the resource is implemented by TSF-internal data structures whose contents are not visible outside the TSF. These resources are completely initialized when created, and have no previous information contents remaining.

Explicit clearing is used only for directory entries, because they are accessible both through TSF interfaces for managing directories and for reading files. Because this exposes the internal structure they must be explicitly cleared when released to prevent the internal state from remaining visible.

Storage management is used in conjunction with explicit initialization for object reuse on files and processes. As memory and file resources are allocated to a process, the kernel keeps track of what has been written (i.e., initialized) and allows read access only to those portions of the applicable objects.

The User Data Protection function is designed to satisfy the following security functional requirements:

- FDP_ACC.1: All operations that a process can perform on the required objects are addressed by the DAC mechanism.
- FDP_ACF.1: The rules defined by permission bits and ACLs are enforced to ensure that non-administrative users are appropriately restricted when accessing controlled objects.
- FDP_RIP.2a: The contents of user objects are protected from inappropriate reuse by initializing contents before access, clearing contents before access, and by limiting accesses to information previously written.
- FDP_RIP.2b: The contents of subjects are protected from inappropriate reuse by initializing contents before access and by limiting accesses to information previously written.

6.1.3 Identification and Authentication

Users can either be individuals (accounts tied to a physical user) or logical (accounts that exist for applications to perform specific tasks). Both types of users have a User ID (UID), a Group ID (GID), and a password. Groups are a mechanism to group users so that, for example, access controls are better organized.

User and group definitions are stored in protected files, modifiable only by administrative users. Passwords are stored in a separate file that is accessible only by administrative users.

When a user attempts to log into the TOE either from a connected terminal or via a network interface, they are required to identify themselves and to provide the applicable password for authentication. Successful authentication requires all of the following to be true:

1. the user has entered a valid user name;
2. the password entered by the user, and encrypted by the TOE, is identical to the encrypted password stored by the TOE for that user;
3. the user account is not locked; and,
4. the password has not expired.

A user account will be locked by the TOE if any of the following are true:

1. the number of consecutive failed login attempts to that user account exceeds the maximum permitted and
2. the user account has been explicitly locked by an administrative user.

An administrative user can define the following restrictions on the authentication process:

1. the maximum number of days for which a password is valid;
2. the minimum acceptable password length;
3. the maximum number of consecutive failed login attempts before a user account is locked; and,
4. whether a user can modify their password.

When a User ID and password are entered the password is hashed and compared with the representation held in the password file for that User ID. Authentication succeeds only if the hashed password matches.

A user may change identity by means of the *su* command. By this means they may adopt, following successful authentication, the real and effective User id and Group id of another user. The most common use of this function would be to adopt an administrative user identity in order to carry out administrative tasks. The administrative user account may also be accessed by means of login, but only at the directly attached console.

A user may also change identity (or group) by executing a program with the setuid (or setgid) bit set. When such a program is executed, the resulting acquires the uid (or gid) assigned to the object by its creator or an authorized administrator. However, it will retain the login ID of the process that executed the program.

A user may also fork additional processes to act on their behalf, in each case all of the security attributes of the new process are copied from the parent process.

Unless prevented by an administrator, users are permitted to change their password through use of the *passwd* command. Users are required to re-authenticate before being permitted to change their passwords, whereas administrative users are not. Initial passwords are set only by an administrative user.

RHEL AS will not echo user passwords to the display when they are entered in the password field. No indication is given on the display that any character has been entered; this protects the password from physical observation.

The Identification and Authentication function is designed to satisfy the following security functional requirements:

- FIA_ATD.1:Users are defined in terms of user IDs, groups IDs, and passwords. Note that the administrative user role is identified with the special user ID of 0.
- FIA_SOS.1: Administrative users have the ability to configure the minimum settable password to ensure that the FIA_SOS.1 requirement is satisfied. Applicable guidance for this setting can be found in the Evaluated Configuration Guide for Red Hat Enterprise Linux.
- FIA_UAU.2: RHEL AS offers no services until a user has successfully logged in.
- FIA_UAU.7: RHEL AS does not echo password as they are entered.
- FIA_UID.2: RHEL AS offers no services until a user has successfully logged in.
- FIA_USB.1: RHEL AS ensures that each users security attributes are assigned to their initial process when they log in and then subsequently ensures that those attributes remain associated with the user's processes appropriately, allowing for changing of the user and group identities in specific and controlled ways.

6.1.4 Security Management

RHEL AS recognizes a single security management role: 'administrative user' (or administrator). Everyone else is considered a 'normal user'. The administrative user role is indicated by a special user ID, 0. A user can log on with this user ID only from the console terminal. Otherwise, any user that knows the administrative user account password can issue the *su* command to assume that role.

There are a number of system calls that require user ID=0 in order to succeed and all of the applicable TSF configuration files, as well as special device files and audit files are restricted, where appropriate, to the administrative user. Among these are user and group definitions files, the password file, audit configuration files, and audit logs. In general, each of the configuration settings becomes effective the next time it is used by the TOE (e.g., next login for user settings, audit restart for audit settings).

In some cases certain configuration data is allowed to be manipulated by normal users. This is accomplished via system calls or setuid programs, which have the necessary access and enforce appropriate controls.

While objects are protected by default when created (by default only the owner/creator has access), users can change object access attributes for objects that the own (or they created in the case of IPC objects only) and users can establish default access permissions using the *umask* setting associated with their process (which serves to mask available permission bits) and for file system objects created within a directory that they own. Note that any change in an object security attribute is effective the next time that attribute is checked (e.g., during an open).

While an administrative user can change any password, normal users can change their own password (unless explicitly restricted from so doing by an administrative user).

The Security Management function is designed to satisfy the following security functional requirements:

- FMT_MSA.1: Once an object is created, only administrative users and creating/owning users can change object security attributes.
- FMT_MSA.3: Objects are protected by default when created, though a user can control the initial access setting using a process *umask* and a default ACL on a parent directory.
- FMT_MTD.1a: Only administrators can access audit data.
- FMT_MTD.1b: Only administrators can access the audit configuration data.
- FMT_MTD.1c: Only administrative users can change the definition of a user (except for passwords, as indicated below).
- FMT_MTD.1d: Only administrative users and the user to which the authentication data belongs can set or change a user password.
- FMT_REV.1a: Only administrative users can revoke user security attributes and those changes are effective the next time the user logs in.
- FMT_REV.1b: Only administrative users and object owners (and object creator in the case of IPC objects only) can change object security attributes and those changes are effective the next time a check is made.
- FMT_SMR.1: RHEL AS differentiates an administrative user from a normal user using the special user ID of 0.

6.1.5 Protection of the TOE Security Functions

RHEL AS protects itself from users primarily through the use of the privilege levels provided by the underlying hardware. The kernel runs on the privileged state while users run in unprivileged state, where resources of the privilege state are inaccessible. The kernel instantiates processes in the unprivileged state in a manner that separates and isolates them. They each are assigned a distinct address space and are allowed to access resources and communicate with each other only by requesting kernel services that are designed to control process operations appropriately. Note that part of the TSF as well as administrative users run in the unprivileged state and are protected from normal users just as normal users are protected from each other.

RHEL AS ensures that its security functions cannot be bypassed by encapsulating all protected resources and offering access only through controlled interfaces (including hardware instructions, kernel or system calls, and communication offered by trusted processes such as daemons).

Both self-protection and non-bypassability also depend substantially on the DAC mechanism. Device objects and TSF data files are all protected so that normal users do not have inappropriate access.

RHEL AS offers diagnostic tools that allow an administrative user to test the memory, CPU, and I/O devices for correct operation or failures on demand.

RHEL AS utilizes clock support provided by the underlying hardware to generate time stamps for use in audit records.

The Protection of the TOE Security Functions function is designed to satisfy the following security functional requirements:

- FPT_AMT.1: Tests are available on demand for the administrative user to test the correct operation of the underlying hardware.
- FPT_RVM.1: RHEL AS has been designed to encapsulate its protected resources and to offer well defined interfaces that ensure that the applicable policies are always enforced.
- FPT_SEP.1: RHEL AS has been designed to protect itself from tampering and to separate users by utilizing hardware support to provide its own execution domain and to separate the processes it instantiates for itself and its users.

- FPT_STM.1: RHEL AS gets reliable time information from the underlying hardware.

6.2 TOE Security Assurance Measures

6.2.1 Configuration management

The configuration management measures applied by Unisys and Red Hat ensure that configuration items are uniquely identified, and that documented procedures are used to control and track changes that are made to the TOE. Unisys and Red Hat ensure changes to the implementation representation are controlled. Unisys and Red Hat perform configuration management on the TOE implementation representation, design documentation, tests and test documentation, user and administrator guidance, delivery and operation documentation, life-cycle documentation, vulnerability analysis documentation, and configuration management documentation.

These activities are documented in:

- Unisys Configuration Management Plan
- Red Hat Enterprise Linux Configuration Management Plan

The Configuration management assurance measure satisfies the following EAL 3 augmented with ALC_FLR.2 assurance requirements:

- ACM_CAP.3
- ACM_SCP.1

6.2.2 Delivery and operation

Unisys and Red Hat provide delivery documentation and procedures to identify the TOE, secure the TOE during delivery, and provide necessary installation and generation instructions. Unisys's and Red Hat's respective delivery procedures describe all applicable procedures to be used to prevent in appropriate access to the TOE. Unisys and Red Hat also provide documentation that describes the steps necessary to install the TOE in accordance with the evaluated configuration.

These activities are documented in:

- Red Hat Enterprise Linux 4 Installation Guide
- Red Hat Enterprise Linux 4 System Administration Guide
- Red Hat Enterprise Linux 4 Security Guide
- Red Hat Enterprise Linux 4 Reference Guide
- Unisys ES7000 Site Planning Guide
- Evaluated Configuration Guide for Red Hat Enterprise Linux

The Delivery and operation assurance measure satisfies the following EAL 3 augmented with ALC_FLR.2 assurance requirements:

- ADO_DEL.1
- ADO_IGS.1

6.2.3 Development

Unisys and Red Hat have numerous documents describing all facets of the design of the TOE. In particular, they have a functional specification that describes the accessible TOE interfaces; a high-level design that decomposes the TOE architecture into subsystems and describes each subsystem and their interfaces; and, correspondence

documentation that explains how each of the design abstractions correspond from the TOE summary specification in the Security Target to the subsystems.

These activities are documented in:

- RHEL3 EAL3 FSP Cross Reference
- RHEL High Level Design
- ES7000 Reference

The Development assurance measure satisfies the following EAL 3 augmented with ALC_FLR.2 assurance requirements:

- ADV_FSP.1
- ADV_HLD.2
- ADV_RCR.1

6.2.4 Guidance documents

Unisys and Red Hat provide administrator and user guidance on how to utilize the TOE security functions and warnings to administrators and users about actions that can compromise the security of the TOE.

These activities are documented in:

- Red Hat Enterprise Linux 4 System Administration Guide
- Red Hat Enterprise Linux 4 Security Guide
- Red Hat Enterprise Linux 4 Reference Guide
- ES7000 Reference

The Guidance documents assurance measure satisfies the following EAL 3 augmented with ALC_FLR.2 assurance requirements:

- AGD_ADM.1
- AGD_USR.1

6.2.5 Life cycle support

Unisys and Red Hat apply security controls on the development environment that are adequate to provide the confidentiality and integrity of the TOE design and implementation that is necessary to ensure the secure development of the TOE. Unisys and Red Hat have procedures that define the process for accepting and acting upon user reports of security flaws. These procedures describe the acceptance criteria for security flaws, how all security flaws and the status of fixes for each security flaw are tracked, and how corrections and corrective measures are made available as applicable.

These activities are documented in:

- Unisys Lifecycle Document
- Red Hat Enterprise Linux Lifecycle Document

The Life cycle support assurance measure satisfies the following EAL 3 augmented with ALC_FLR.2 assurance requirements:

- ALC_DVS.1
- ALC_FLR.2

6.2.6 Tests

Unisys and Red Hat have test plans that describe how each of the necessary security functions is tested, along with the expected test results. Unisys and Red Hat have documented each test as well as an analysis of test coverage and depth demonstrating that the security aspects of the design evident from the functional specification and high-level design are appropriately tested. Actual test results are created on a regular basis to demonstrate that the tests have been applied and that the TOE operates as designed.

These activities are documented in:

- Red Hat Enterprise Linux w/Unisys ES7000 Test Plan
- Red Hat Enterprise Linux Test Coverage Analysis
- Red Hat Enterprise Linux w/Unisys ES7000 Test Results

The Tests assurance measure satisfies the following EAL 3 augmented with ALC_FLR.2 assurance requirements:

- ATE_COV.2
- ATE_DPT.1
- ATE_FUN.1
- ATE_IND.2

6.2.7 Vulnerability assessment

The TOE administrator and user guidance documents describe the operation of RHEL AS and how to maintain a secure state. These guides also describe all necessary operating assumptions and security requirements outside the scope of control of the TOE. They have been developed to serve as complete, clear, consistent, and reasonable administrator and user references.

Unisys and Red Hat have conducted a strength of function analysis wherein all permutational or probabilistic security mechanisms have been identified and analyzed resulting in a demonstration that all of the relevant mechanisms fulfill the minimum strength of function claim, SOF-Medium.

Unisys and Red Hat perform regular vulnerability analyses of the entire TOE (including documentation) to identify obvious weaknesses that can be exploited in the TOE.

These activities are documented in:

- Red Hat Enterprise Linux Vulnerability Analysis

The Vulnerability assessment assurance measure satisfies the following EAL 3 augmented with ALC_FLR.2 assurance requirements:

- AVA_MSU.1
- AVA_SOF.1
- AVA_VLA.1

7.0 Protection Profile Claims

This Security Target is conformant with the Controlled Access Protection Profile (CAPP) Version 1.d, 8 October 1999 (CAPP).

The environment statements and security objectives have all been copied from the CAPP. Each of the security requirements have been copied from the CAPP with the following modifications:

- FAU_STG.1 has been updated in accordance with International Interpretations that serve to qualify that the necessary protection is of the audit records in the audit trail.
- FIA_UAU.1 has been replaced with the hierarchically greater FIA_UAU.2 from the Common Criteria.
- FIA_UID.1 has also been replaced with the hierarchically greater FIA_UID.2 from the Common Criteria.
- FIA_USB.1 has been updated to indicate that International Interpretations have been applied. However, the International Interpretation matches the refinements already made in the CAPP and as a result the requirement is not effectively changed.
- The assurance requirements have been replaced with the current Internationally Interpreted EAL 3 augmented with ALC_FLR.2 representations which are equivalent or greater than the CAPP assurance requirements.

8.0 Rationale

This section provides the rationale for completeness and consistency of the Security Target. The rationale addresses the following areas:

- Security Objectives;
- Security Functional Requirements;
- Security Assurance Requirements;
- Strength of Functions;
- Requirement Dependencies;
- TOE Summary Specification; and,
- PP Claims.

8.1 Security Objectives Rationale

All of the security environment statements and security objectives have been drawn from the Controlled Access Protection Profile (CAPP) Version 1.d, 8 October 1999 (CAPP). The relevant rationale can be found in the CAPP.

8.2 Security Requirements Rationale

All of the security objectives and security functional requirements have been drawn from the Controlled Access Protection Profile (CAPP) Version 1.d, 8 October 1999 (CAPP), with the exception of FIA_UAU.2 and FIA_UID.2. FIA_UAU.2 is hierarchical to and replaces FIA_UAU.1 from the CAPP and FIA_UID.2 is hierarchical to and replaces FIA_UID.1 from the CAPP. In each case, the rationale in the CAPP should still apply since these replacements represent a stronger version of the requirements they replace. Hence, the necessary rationale can be found in the CAPP.

8.3 Security Assurance Requirements Rationale

In generalt, the Controlled Access Protection Profile (CAPP) Version 1.d, 8 October 1999 (CAPP) provides the necessary rationale for the selected assurance requirements. The EAL 3 augmented with ALC_FLR.2 assurance target has been selected in this Security Target to conform with the CAPP, but also to exceed its assurance objectives in order to ensure that identified flaws are addressed

8.4 Strength of Functions Rationale

EAL3 augmented with ALC_FLR.2 was selected as the assurance level because the TOE is a commercial product whose users require a moderate to high level of independently assured security, and the vendor has implemented procedures to accept and act upon user reports of security flaws. The TOE is targeted at a relatively benign environment with good physical access security and competent administrators. Within such environments it is assumed that attackers will have little attack potential. As such, EAL3 augmented with ALC_FLR.2 is appropriate to provide the assurance necessary to counter the limited potential for attack.

8.5 Requirement Dependency Rationale

As indicated in the table below, all of the dependencies defined in the Common Criteria have been addressed in this Security Target, except for FMT_SMF.1 (identified in **bold red**). Note that FIA_UID.1 was replaced by FIA_UID.2 and FIA_UAU.1 was replaced by FIA_UAU.2, but the corresponding dependencies are unchanged.

Subsequent to the publication and validation of the Controlled Access Protection Profile (CAPP) Version 1.d, 8 October 1999 (CAPP), International Interpretation #65 was finalized. This interpretation introduces a new family of Security Management requirements, Specification of Management Functions (FMT_SMF). While this should not normally affect dependency rationale, that interpretation introduces dependencies from FMT_MSA.1 and FMT_MTD.1, both contained in this Security Target. Hence, it seems as though some FMT_SMF security requirements should be added to this Security Target to fulfill those dependencies. However, while the CAPP is clearly intended to ensure that certain security management functions are controlled if they are made available, it is not evident from the CAPP which, if any, of those security management functions must be present in the first place. This Security Target identifies all applicable security management functions in the TOE and explains how they are appropriately controlled and it is effectively unnecessary to introduce a security functional requirement to demand that certain security management functions must be present.

ST Requirement	CC Dependencies	ST Dependencies
FAU_GEN.1	FPT_STM.1	FPT_STM.1
FAU_GEN.2	FAU_GEN.1 and FIA_UID.1	FAU_GEN.1 and FIA_UID.2
FAU_SAR.1	FAU_GEN.1	FAU_GEN.1
FAU_SAR.2	FAU_SAR.1	FAU_SAR.1
FAU_SAR.3	FAU_SAR.1	FAU_SAR.1
FAU_SEL.1	FAU_GEN.1 and FMT_MTD.1	FAU_GEN.1 and FMT_MTD.1b
FAU_STG.1	FAU_GEN.1	FAU_GEN.1
FAU_STG.3	FAU_STG.1	FAU_STG.1
FAU_STG.4	FAU_STG.1	FAU_STG.1
FDP_ACC.1	FDP_ACF.1	FDP_ACF.1
FDP_ACF.1	FDP_ACC.1 and FMT_MSA.3	FDP_ACC.1 and FMT_MSA.3
FDP_RIP.2a	none	none
FDP_RIP.2b	none	none
FIA_ATD.1	none	none
FIA_SOS.1	none	none
FIA_UAU.2	FIA_UID.1	FIA_UID.2
FIA_UAU.7	FIA_UAU.1	FIA_UAU.2
FIA_UID.2	none	none
FIA_USB.1	FIA_ATD.1	FIA_ATD.1
FMT_MSA.1	FMT_SMR.1 and FMT_SMF.1 and (FDP_ACC.1 or FDP_IFC.1)	FMT_SMR.1 and **[FMT_SMF.1]** and FDP_ACC.1
FMT_MSA.3	FMT_MSA.1 and FMT_SMR.1	FMT_MSA.1 and FMT_SMR.1
FMT_MTD.1a	FMT_SMR.1 and FMT_SMF.1	FMT_SMR.1 and **[FMT_SMF.1]**
FMT_MTD.1b	FMT_SMR.1 and FMT_SMF.1	FMT_SMR.1 and **[FMT_SMF.1]**
FMT_MTD.1c	FMT_SMR.1 and FMT_SMF.1	FMT_SMR.1 and **[FMT_SMF.1]**
FMT_MTD.1d	FMT_SMR.1 and FMT_SMF.1	FMT_SMR.1 and **[FMT_SMF.1]**
FMT_REV.1a	FMT_SMR.1	FMT_SMR.1
FMT_REV.1b	FMT_SMR.1	FMT_SMR.1
FMT_SMR.1	FIA_UID.1	FIA_UID.2
FPT_AMT.1	none	none
FPT_RVM.1	none	none
FPT_SEP.1	none	none
FPT_STM.1	none	none
ACM_CAP.3	ALC_DVS.1	ALC_DVS.1
ACM_SCP.1	ACM_CAP.3	ACM_CAP.3
ADO_DEL.1	none	none
ADO_IGS.1	AGD_ADM.1	AGD_ADM.1
ADV_FSP.1	ADV_RCR.1	ADV_RCR.1
ADV_HLD.2	ADV_FSP.1 and ADV_RCR.1	ADV_FSP.1 and ADV_RCR.1
ADV_RCR.1	none	none
AGD_ADM.1	ADV_FSP.1	ADV_FSP.1

AGD_USR.1	ADV_FSP.1	ADV_FSP.1
ALC_DVS.1	none	none
ALC_FLR.2	none	none
ATE_COV.2	ADV_FSP.1 and ATE_FUN.1	ADV_FSP.1 and ATE_FUN.1
ATE_DPT.1	ADV_HLD.1 and ATE_FUN.1	ADV_HLD.2 and ATE_FUN.1
ATE_FUN.1	none	none
ATE_IND.2	ADV_FSP.1 and AGD_ADM.1 and AGD_USR.1 and ATE_FUN.1	ADV_FSP.1 and AGD_ADM.1 and AGD_USR.1 and ATE_FUN.1
AVA_MSU.1	ADO_IGS.1 and ADV_FSP.1 and AGD_ADM.1 and AGD_USR.1	ADO_IGS.1 and ADV_FSP.1 and AGD_ADM.1 and AGD_USR.1
AVA_SOF.1	ADV_FSP.1 and ADV_HLD.1	ADV_FSP.1 and ADV_HLD.2
AVA_VLA.1	ADV_FSP.1 and ADV_HLD.1 and AGD_ADM.1 and AGD_USR.1	ADV_FSP.1 and ADV_HLD.2 and AGD_ADM.1 and AGD_USR.1

8.6 Explicitly Stated Requirements Rationale

All of the requirements in this Security Target have been drawn from the Controlled Access Protection Profile (CAPP) Version 1.d, 8 October 1999 (CAPP) and applicable International Interpretations (except where FIA_UAU.1 was replaced by FIA_UAU.2 and FIA_UID.1 was replaced by FIA_UID.2, both from the Common Criteria Part 2 version 2.2 revision 256). As such, no explicitly stated requirements have been defined in this Security Target. Note that if there are any explicitly stated requirements in the CAPP, the CAPP should provide the necessary rationale.

8.7 TOE Summary Specification Rationale

Each subsection in Section 6, the TOE Summary Specification, describes a security function of the TOE. Each description is followed with rationale that indicates which requirements are satisfied by aspects of the corresponding security function. The set of security functions work together to satisfy all of the security functions and assurance requirements. Furthermore, all of the security functions are necessary in order for the TSF to provide the required security functionality.

This Section in conjunction with Section 6, the TOE Summary Specification, provides evidence that the security functions are suitable to meet the TOE security requirements. The collection of security functions work together to provide all of the security requirements. The security functions described in the TOE summary specification are all necessary for the required security functionality in the TSF. **Table 3 Security Functions vs. Requirements Mapping** demonstrates the relationship between security requirements and security functions.

	Security Audit	**User Data Protection**	**Identification and Authentication**	**Security Management**	**Protection of the TOE Security Functions**

FAU_GEN.1	X				
FAU_GEN.2	X				
FAU_SAR.1	X				
FAU_SAR.2	X				
FAU_SAR.3	X				
FAU_SEL.1	X				
FAU_STG.1	X				
FAU_STG.3	X				
FAU_STG.4	X				
FDP_ACC.1		X			
FDP_ACF.1		X			
FDP_RIP.2a		X			
FDP_RIP.2b		X			
FIA_ATD.1			X		
FIA_SOS.1			X		
FIA_UAU.2			X		
FIA_UAU.7			X		
FIA_UID.2			X		
FIA_USB.1			X		
FMT_MSA.1				X	
FMT_MSA.3				X	
FMT_MTD.1a				X	
FMT_MTD.1b				X	
FMT_MTD.1c				X	
FMT_MTD.1d				X	
FMT_REV.1a				X	
FMT_REV.1b				X	
FMT_SMR.1				X	
FPT_AMT.1					X
FPT_RVM.1					X
FPT_SEP.1					X
FPT_STM.1					X

Table 3 Security Functions vs. Requirements Mapping

8.8 PP Claims Rationale

See Section 7, Protection Profile Claims.