1.

Apple Computer
Mac OS X v10.3.6 and
Mac OS X Server v10.3.6
Security Target

Version 1.0
December 13, 2004

Prepared for:
Apple Computer, Inc.
1 Infinite Loop
Cupertino, CA 95014

Prepared By:
Science Applications International Corporation
Common Criteria Testing Laboratory
7125 Gateway Drive, Suite 300
Columbia, MD 21046

TABLE OF CONTENTS

1. Security Target Introduction

This section contains document management and overview information. This section identifies the Security Target (ST) and Target of Evaluation (TOE), ST conventions, ST conformance claims, and the ST organization.

1.1 Security Target, TOE and CC Identification

ST Title – Apple Computer Apple Computer Mac OS X v10.3.6 and Mac OS X Server v10.3.6 Security Target

ST Version – Version 1.0

ST Date – December 13, 2004

TOE Software Identification – Mac OS X v10.3.6 and Mac OS X Server v10.3.6 with Common Criteria Tools Package

TOE Hardware Identification – The following hardware platforms are included in the evaluated configuration:

- Mac OS X version 10.3.6
  - eMac G4
  - iMac G3
  - iMac G4
  - iMac G5
  - iBook G3
  - iBook G4
  - PowerBook G3
  - PowerBook G4
  - Power Mac G3
  - Power Mac G4 Cube
  - Power Mac G4 (single processor)
  - Power Mac G4 Dual Processor
  - Power Mac G5 (single processor)
  - Power Mac G5 Dual Processor
- Mac OS X Server version 10.3.6
  - Power Mac G4 (single processor)
  - Power Mac G4 Dual Processor
  - Power Mac G5 (single processor)
  - Power Mac G5 Dual Processor
  - Xserve G4 (single processor)
  - Xserve G4 Dual Processor

- Xserve G5 (single processor)
- Xserve G5 Dual Processor

PP Identification – Controlled Access Protection Profile (CAPP), version 1.d, October 8, 1999

CC Identification – Common Criteria for Information Technology Security Evaluation, Version 2.1, August 1999, ISO/IEC 15408.

Keywords – operating system, discretionary access control, audit, identification, and authentication

1.2 CC Conformance Claims

This TOE conforms to the following specifications:

- Common Criteria for Information Technology Security Evaluation Part 2: Security functional requirements, Version 2.1, August 1999, ISO/IEC 15408-2.
  - Part 2 conformant
- Common Criteria for Information Technology Security Evaluation Part 3: Security assurance requirements, Version 2.1, August 1999, ISO/IEC 15408-3.
  - Part 3 conformant
  - Evaluation Assurance Level 3 (EAL 3)
- Controlled Access Protection Profile (CAPP), version 1.d, October 8, 1999.

1.3 Strength of Environment

Mac OS X provides a moderate level of independently assured security in a conventional TOE and is suitable for a cooperative non-hostile environment. The assurance requirements and the minimum strength of function were chosen to be consistent with this goal and to be compliant with the Controlled Access Protection Profile (CAPP). The TOE assurance level is Evaluation Assurance Level (EAL) 3 and the TOE minimum strength of function is SOF-medium.

1.4 Conventions, Terminology, Acronyms

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

1.4.1 Conventions

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.1 (a) and FDP_ACC.1 (b) indicate that the ST includes two iterations of the FDP_ACC.1 requirement, a and b.
  - Assignment: allows the specification of an identified parameter.
  - Selection: allows the specification of one or more elements from a list.
  - Refinement: allows the addition of details.

  The conventions for the operations are described in section 5.

- Other sections of the ST – Other sections of the ST use bolding to highlight text of special interest, such as captions.

1.4.2 Terminology

This section describes terms that are used throughout the ST. When possible, terms are defined, as they exist in the *Common Criteria for Information Technology Security Evaluation*:

- Authorized administrator / Administrator – A user in the administrator role is an authorized user who has been granted the authority to manage the TOE. These users are expected to use this authority only in the manner prescribed by the guidance given them. The term authorized administrator is taken from the CC and CAPP and is used in the ST in those sections that are derived from the CAPP or the CC directly. Otherwise, the term administrator is used. These terms are used interchangeably.
- Authorized User – A user who has been properly identified and authenticated. These users are considered to be legitimate users of the TOE.
- Discretionary Access Control Policy (DAC) – A policy that allows authorized users and authorized administrators to control access to objects on the basis of individual user identity or membership in a group
- Non-kernel Objects – Objects that are managed by trusted processes in user-mode.
- Protection Profile (PP) - An implementation-independent set of security requirements for a category of TOEs that meet specific consumer needs.
- Security Target (ST) - A set of security requirements and specifications to be used as the basis for evaluation of an identified TOE.
- Target of Evaluation (TOE) - An IT product of system and its associated administrator and user guidance documentation that is the subject of an evaluation.
- TOE Security Functions (TSF) - A set consisting of all hardware, software, and firmware of the TOE that must be relied upon for the correct enforcement of the TSP.
- TOE Security Policy (TSP) - A set of rules that regulate how assets are managed, protected, and distributed within a TOE.
- TSF data - Data created by and for the TOE that might affect the operation of the TOE.
- TSF Scope of Control (TSC) - The set of interactions that can occur with or within a TOE and are subject to the rules of the TSP.
- User – An individual who attempts to invoke a service offered by the TOE.

1.4.3 Acronyms

The acronyms used in this Security Target are specified in Appendix A – List of Acronyms

1.5 Security Target Overview and Organization

The Mac OS X Target of Evaluation (TOE) is a networked, general-purpose operating system. Mac OS X and Mac OS X Server enforce the same security functions; the only differences lie in the area of performance. Hence both will be referred to simply as Mac OS X throughout the ST. Mac OS X is a fully functioning Unix operating system, based on the Mach kernel and FreeBSD, which abstracts the complexity of Unix and provides a user interface that fosters enhanced productivity and ease of use. The Mac OS X TOE provides the following security services: audit, user data protection, identification and authentication, security management, and protection of the TOE Security Functions (TSF).

The Mac OS X Security Target contains the following additional sections:

- TOE Description (Section 2)
- Security Environment (Section 3)
- Security Objectives (Section 4)
- IT Security Requirements (Section 5)
- TOE Summary Specification (Section 6)
- Protection Profile Claims (Section 7)
- Rationale (Section 8).

2. TOE Description

The TOE includes the Mac OS X operating system, supporting hardware, and those applications necessary to manage, support and configure the operating system. The TOE is a subset of the Mac OS X product as defined in the administrator guidance. Apple provides several Mac OS X software applications that are considered outside the scope of the defined TOE and thus not part of the evaluated configuration. Services outside this evaluation include: e-mail services; web server services; remote apple events; print sharing services; file sharing services; and classic programming support (Old Macintosh OS compatibility support). Mac OS X Server contains a watchdog timer to restart services and provide stability. However, this timer is disabled in the evaluated configuration.

Mac OS X is a completely rebuilt implementation of the Macintosh operating system, offering stability, power and interoperability, beneath an elegant user interface. Built upon an open source, UNIX-based core called Darwin, Mac OS X lays the groundwork for a new generation of developer innovation. The core of Mac OS X, Darwin, is an open source project. Darwin integrates a number of technologies, including the Mach 3.0 kernel, operating system services based on BSD UNIX (Berkeley Software Distribution), high-performance networking facilities, and support for multiple integrated file systems. Further, Darwin's modular design lets developers dynamically load such things as device drivers, networking extensions, and new file systems.

Darwin provides an advanced memory protection and management system. Darwin ensures reliability by protecting applications with a robust architecture that allocates a unique address space for each application or process. The Mach kernel augments standard virtual memory semantics with the abstraction of memory objects. This enables Mac OS X to manage separate application environments simultaneously.

Device drivers are created using an object-oriented programming framework called I/O Kit. Drivers created with I/O Kit easily acquire true plug and play, dynamic device management ("hot plugging"), and power management. I/O kit also provides hardware access to high-level application software. For network protocol developers, Darwin provides the Network Kernel Extension (NKE) facility. This allows developers to create networking modules and even entire protocol stacks that can be dynamically loaded and unloaded. NKEs also make it possible to configure protocol stacks automatically and easily monitor and modify network traffic. At the data-link and network layers, they can also receive notifications of asynchronous events from device drivers.

Although Darwin also offers support for multiple file systems, in the evaluated configuration, only the HFS+ filesystem is supported. Darwin also supplies the following advanced functionality:

- Preemptive and cooperative multitasking via the Mach kernel
- Symmetric multiprocessing (SMP) augmented by support for multithreading
- Real-time support guaranteeing low-latency access to processor resources for time-sensitive media applications.

Mac OS X supports a wide range of protocols and network services. In the evaluated configuration, the TCP/IP protocol and the NFS, DNS, and SSH services are supported.

2.1 Product Description

Mac OS X is a fully functioning Unix operating system, based on the Mach kernel and FreeBSD, which abstracts the complexity of Unix and provides a user interface that fosters enhanced productivity and ease of use. Figure 1 provides an overview of the architecture of Mac OS X.

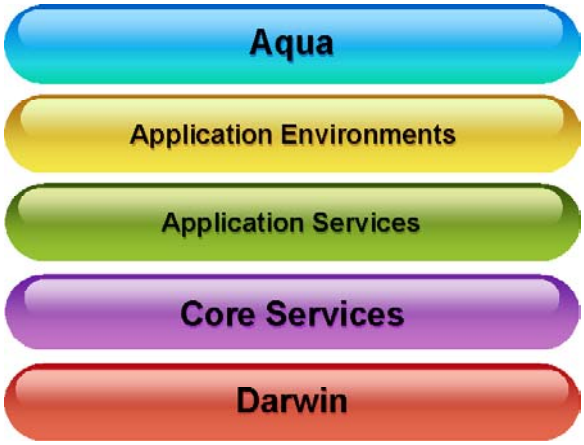

Figure 1 Mac OS X Architecture

The system components perform the following functions:

- Aqua – This provides the graphical interface to both users and administrators.
- Application Environments – This layer provides an application environment for users. An application environment consists of the frameworks, libraries, and services (along with associated APIs) necessary for the runtime execution of programs developed with those APIs. The application environments have dependencies on all underlying layers of system software. There are five application environments provided:
  1. Carbon – a set of programming interfaces derived from earlier Mac OS APIs that have been modified to work with Mac OS X, especially its kernel environment.
  2. Cocoa – a native set of Mac OS X APIs that access Mac OS X features using an object framework.
  3. Classic - provides a compatibility environment that allows legacy applications to execute on Mac OS X. The Classic environment is essentially a virtual machine that allows legacy applications to execute in a simulated environment where older APIs are translated to Mac OS X APIs. This environment is not included in the evaluated configuration since older versions of the operating system are not included in the evaluation.
  4. Java – provides development and runtime environments and an application framework that allow applications developed in Java to execute on Mac OS X.
  5. BSD Commands – a native implementation of a command line BSD command environment.
- Application Services -This layer contains the graphics and windowing environment of Mac OS X. This environment is responsible for screen rendering, printing, event handling, and low-level window and cursor management. It also holds libraries, frameworks, and background servers useful in the implementation of graphical user interfaces.
- Core Services - The Core Services layer includes a number of Carbon managers that offer low-level services to all application environments. These services include cooperative and preemptive threading, resource management, memory management, and file-system operations.
- Darwin - The kernel environment is the lowest layer of system software. The kernel environment provides essential operating-system functionality to the layers above it, such as:
  - Preemptive multitasking
  - Advanced virtual memory with memory protection and dynamic memory allocation
  - Symmetric multiprocessing
  - Multi-user access
  - File systems based on VFS (Virtual File System)
  - Device drivers
  - Networking
  - Basic threading packages.

2.2 Security Environment TOE Boundary

The TOE includes both physical and logical boundaries. This section describes both the security functions provided by the TOE as well as the physical realization of the TOE.

2.2.1 Logical Boundaries

Mac OS X provides a rich set of security functions. Each security function is identified and described below:

- Audit – Mac OS has the ability to audit user actions and store the records in an audit trail that is protected from unauthorized access. The administrator has the ability to select which events get audited and to sort and search the audit log after the records have been collected. The audit facility is flexible in permitting the administrator the ability to decide if the system should overwrite old records or halt if the audit trail becomes full.
- User Data Protection – Mac OS X provides a discretionary access control mechanism to protect user objects such as files, directories, and message queues. Access to these objects is mediated by the operating system and granted only if a set of rules is passed. In addition to controlling access, Mac OS X ensures that whenever a user is allocated a resource, that resource is clear of any previous information that it may have contained.
- Identification and Authentication – All users on Mac OS X are identified and authenticated before they can access any system service. Mac OS X maintains a user database with user name, group associations, and authentication information. Mac OS X supports password authentication in the evaluated configuration..
- Security Management – Mac OS X provides a rich set of administrative functions. Graphical tools are provided to manage user accounts, object access rights, and the audit trail.
- TOE Self Protection - Mac OS X has several features to protect the security functions. Mac OS X utilizes the security features of the hardware including running the kernel in the most privileged state of the hardware. Memory protection and process isolation are provided to keep processes from interfering with each other and, more importantly, from interfering with the operating system. There is also a set of diagnostic tools provided to the administrator that can be run to ensure the correct operation of the hardware.

2.2.2 Physical Boundaries

The evaluated hardware is one or more laptops, desktops, and servers connected via an Ethernet with one or more PowerPC G3, G4 or G5 processors running Mac OS X. A set of devices may be attached and they are listed as follows:

- Display Monitor
- Keyboard
- Mouse
- CD-ROM Drive
- Fixed Disk Drives
- Printer
- Audio Adaptor
- Network Adaptor.

The TOE does not include any physical network components between network adaptors of a connection. The ST assumes that any network connections, equipment, and cables are appropriately protected in the TOE security environment.

3. Security Environment

The TOE security environment consists of the threats to security, organizational security policies, and usage assumptions as they relate to Mac OS X. This section describes the security aspects of the environment in which the TOE is intended to be used and the manner in which it is expected to be employed. The material for the environmental description was taken from the CAPP.

3.1 Threats to Security

The CAPP has derived all security objectives from the statement of Organizational Security Policy found in the following section. Therefore, there is no statement of the explicit threats countered by the CAPP.

3.2 Organization Security Policies

An Organizational Security Policy is a set of rules or procedures imposed by an organization upon its operations to protect its sensitive data. Although the organizational security policies described below are drawn from DoD Manual 5200.28-M (Techniques and procedures for Implementing, Deactivating and Evaluating Resource Sharing ADP Systems) they apply to many non-DoD environments.

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.

P.ACCOUNTABILITY

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

3.3 Secure Usage Assumptions

This section contains assumptions regarding the security environment and the intended usage of the TOE. The TOE is assured to provide effective security measures in a cooperative non-hostile environment only if it is installed, managed, and used correctly. The operational environment must be managed in accordance with assurance requirements documentation for delivery, operation, and user/administrator guidance. The following specific conditions are assumed to exist in an environment where the TOE is employed.

3.3.1 Physical Assumptions

The TOE is intended for application in user areas that have physical control and monitoring. It is assumed that the following physical conditions will exist:

A.LOCATE

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

A.PROTECT

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

3.3.2 Personnel Assumptions

It is assumed that the following personnel conditions will exist:

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.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.

3.3.3 Connectivity Assumptions

The CAPP contains no explicit network or distributed system requirements. However, it is assumed that the following connectivity conditions exist:

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.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.

4. Security Objectives

This section defines the security objectives of Mac OS X and its supporting environment. Security objectives reflect the stated intent to counter identified threats and/or comply with any organizational security policies identified. All of the identified threats and organizational policies are addressed under one of the categories below. All of the objectives were taken directly from the CAPP without modifications.

4.1 IT Security Objectives

The following are the TOE security objectives:

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.AUDITING

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

O.RESIDUAL_INFORMATION

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

O.MANAGE

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

O.ENFORCEMENT

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

4.2 TOE Non-IT Security Objectives

The TOE's general operating environment must satisfy the following objectives. These objectives do not levy any IT requirements but are satisfied by procedural or administrative measures.

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.

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.

5. IT Security Requirements

5.1 TOE Security Functional Requirements

This section specifies the security functional requirements (SFRs) for the TOE. Functional requirements components in this ST were drawn from the CAPP. The CAPP uses Part 2 of the CC. Some functional requirements are extensions to those found in the CC. This section organizes the SFRs by CC class. The functional security requirements for the ST consist of the following SFRs, summarized in Table 1 Security Functional Requirements.

Security Functional Class	Security Functional Components
Security Audit (FAU)	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
User Data Protection (FDP)	FDP_ACC.1 Discretionary Access Control Policy
	FDP_ACF.1 Discretionary Access Control Functions
	FDP_RIP.2 Object Residual Information Protection
	Note 1 Subject Residual Information Protection
Identification and authentication (FIA)	FIA_ATD.1 User attribute definition
	FIA_SOS.1 Strength of authentication data
	FIA_UAU.1 Timing of authentication
	FIA_UAU.7 Protected Authentication Feedback
	FIA_UID.1 Timing of identification
	FIA_USB.1 User-subject binding
Security management (FMT)	FMT_MSA.1 Management of object security attributes
	FMT_MSA.3 Static attribute initialization
	FMT_MTD.1 (a) Management of the audit trail
	FMT_MTD.1 (b) Management of audited events
	FMT_MTD.1 (c) Management of user attributes
	FMT_MTD.1 (d) Management of authentication data
	FMT_REV.1 (a) Revocation of user attributes

Security Functional Class	Security Functional Components
	FMT_REV.1 (b) Revocation of object attributes
	FMT_SMR.1Security roles
Protection of the TSF (FPT)	FPT_AMT.1Abstract machine testing
	FPT_RVM.1Non-bypassability of the TSP
	FPT_SEP.1TSF domain separation
	FPT_STM.1Reliable time stamps

Table 1 Security Functional Requirements

The CC permits four functional component operations—assignment, refinement, selection, and iteration — to be performed on security functional requirements. This section highlights the four operations in the following manner:

- Assignment: allows the specification of an identified parameter. Indicated with underlined text;
- Refinement: allows the addition of details. Indicated with italics text;
- Selection: allows the specification of one or more elements from a list. Indicated with underlined text; and
- Iteration: allows a component to be used more than once with varying operations. Indicated by a letter in parenthesis placed at the end of the component.

5.1.1 Security audit (FAU)

5.1.1.1 Audit data generation (FAU_GEN.1)

5.1.1.1.1 FAU_GEN.1.1

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

Section	Component	Event	Details
5.1.1	FAU_GEN.1	Start-up and shutdown of audit functions	
5.1.2	FAU_GEN.2	None	
5.1.3	FAU_SAR.1	Reading of information from the audit records.	
5.1.4	FAU_SAR.2	Unsuccessful attempts to read information from the audit records.	
5.1.5	FAU_SAR.3	None	
5.1.6	FAU_SEL.1	All modifications to the audit configuration that occur while the audit collection functions are operating.	
5.1.7	FAU_STG.2	None	
5.1.8	FAU_STG.3.	Actions taken due to exceeding of a threshold	
5.1.9	FAU_STG.4	Actions taken due to the audit storage failure.	
5.2.1	FDP_ACC.1	None	
5.2.2	FDP_ACF.1	All requests to perform an operation on an object covered by the SFP.	The identity of the object.
5.2.3	FDP_RIP.2	None	
5.2.4	Note 1	None	

Section	Component	Event	Details
5.3.1	FIA_ATD.1	None	
5.3.2	FIA_SOS.1	Rejection or acceptance by the TSF of any tested secret.	
5.3.3	FIA_UAU.1	All use of the authentication mechanism.	
5.3.4	FIA_UAU.7	None	
5.3.5	FIA_UID.1	All use of the user identification mechanism, including the identity provided during successful attempts.[1]	The origin of the attempt (e.g. terminal identification.)
5.3.6	FIA_USB.1	Success and failure of binding user security attributes to a subject (e.g. success and failure to create a subject).	
5.4.1	FMT_MSA.1	All modifications of the values of security attributes.	
5.4.2	FMT_MSA.3	Modifications of the default setting of permissive or restrictive rules. All modifications of the initial value of security attributes.	
5.4.3	FMT_MTD.1 (a)	All modifications to the values of TSF data.	
5.4.4	FMT_MTD.1 (b)	All modifications to the values of TSF data.	The new value of the TSF data.
5.4.5	FMT_MTD.1 (c)	All modifications to the values of TSF data.	The new value of the TSF data.
5.4.6	FMT_MTD.1 (d)	All modifications to the values of TSF data.	
5.4.7	FMT_REV.1 (a)	All attempts to revoke security attributes.	
5.4.8	FMT_REV.1 (b)	All modifications to the values of TSF data.	
5.4.9	FMT_SMR.1	Modifications to the group of users that are part of a role.	
5.4.10	FMT_SMR.1	Every use of the rights of a role. (Additional / Detailed)	The role and the origin of the request.
5.5.1	FPT_AMT.1.	Execution of the tests of the underlying machine and the results of the test.	
5.5.2	FPT_RVM.1	None	
5.5.3	FPT_SEP.1	None	
5.5.4	FPT_STM.1	Changes to the time.	

Table 2 Auditable Events

5.1.1.1.2 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; and

b. The additional information specified in the Details column of *Table 2 Auditable Events.*[2]

[1] This meets the interpretation I-410 as it only requires auditing of the subject identity on successful login attempts.
[2] The FAU_GEN.1 requirement derived from the CAPP is a valid refinement of I-0407 as all elements are addressed as deined in the interpretation.

5.1.1.2 User Identity Association (FAU_GEN.2)

5.1.1.2.1 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)

5.1.1.3.1 FAU_SAR.1.1

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

5.1.1.3.2 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)

5.1.1.4.1 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)

5.1.1.5.1 FAU_SAR.3.1

The TSF shall provide the ability to perform searches and sorting of audit data based on the following attributes:

a. User identity.

5.1.1.6 Selective audit (FAU_SEL.1)

5.1.1.6.1 FAU_SEL.1

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

a. User identity,

b. Success and failure and

c. No additional attributes.

5.1.1.7 Protected audit trail storage (FAU_STG.1)

5.1.1.7.1 FAU_STG.1.1

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

5.1.1.7.2 FAU_STG.1.2

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

5.1.1.8 Action in case of possible audit data loss (FAU_STG.3)

5.1.1.8.1 FAU_STG.3.1

The TSF shall generate an alarm to the authorized administrator if the audit trail exceeds 80% capacity.

5.1.1.9 Prevention of audit data loss (FAU_STG.4)

5.1.1.9.1 FAU_STG.4

The TSF shall *be able to* prevent auditable events, except those taken by the authorized administrator, and overwrite old records if the audit trail is full.

5.1.2 User Data Protection

5.1.2.1 Discretionary Access Control Policy (FDP_ACC.1)

5.1.2.1.1 FDP_ACC.1.1

The TSF shall enforce the Discretionary Access Control Policy on subjects: processes acting on the behalf of users, objects: files, directories, named pipes, symbolic links, unnamed pipes, shared memory segment, processes, SysV/Posix semaphores, notifications, BSD locks, at jobs, crontab files, and print queue entries, and all operations among subjects and objects covered by the DAC policy.

5.1.2.2 Discretionary Access Control Functions (FDP_ACF.1)

5.1.2.2.1 FDP_ACF.1.1[3]

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:

i) Unix Permission bits,

ii) Object Ownership,

iii) Object Creator, and

iv) Owning Group.

5.1.2.2.2 FDP_ACF.1.2

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

[3] FDP_ACF.1.1 is copied directly from the CAPP. The text from the CAPP address the I-0418 interpretation as it clearly associated subject attributes with subjects and object attributes with objects.

1. If an object has permission bits, access is granted if one of the following is true:
    a. If the subject is the owner of the object and the object's owner Unix permission bits indicate that the operation required accesses are allowed.
    b. If the subject is a member of the object owning group and the object's group Unix permission bits indicate that the operation required accesses are allowed, or
    c. If the object's world Unix permission bits indicate that the operation required accesses are allowed.
2. The subject's effective or real UID is the same as the object owner or creator and the operation is performed on a process, SysV/Posix semaphore, notification, BSD lock, at job, crontab file, or print queue entry
3. The subject's process ID is the same as the object owner or creator and the operation is performed on a process.

5.1.2.2.3 FDP_ACF.1.3

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

1. If a subject has the UID of 0, the TSF shall authorize access of the subject to any object, even if such access is disallowed by FDP_ACF.1.2.
2. If a subject is in the administrator group, the TSF shall authorize access of the subject to any file object, even if such access is disallowed by FDP_ACF.1.2.

5.1.2.2.4 FDP_ACF.1.4

The TSF shall explicitly deny access of subjects to objects based on no additional explicit denial rules.

5.1.2.3 Object Residual Information Protection (FDP_RIP.2)

5.1.2.3.1 FDP_RIP.2.1

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

5.1.2.4 Subject Residual Information Protection (Note 1)

5.1.2.4.1 Note 1

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

5.1.3 Identification and Authentication (FIA)

5.1.3.1.1 User attribute definition (FIA_ATD.1)

5.1.3.1.2 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, and

d. Security-relevant roles.

5.1.3.2 Verification of Secrets (FIA_SOS.1)

5.1.3.2.1 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 Timing of authentication (FIA_UAU.1)

5.1.3.3.1 FIA_UAU.1.1

The TSF shall allow no function on behalf of the user to be performed before the user is authenticated.

5.1.3.3.2 FIA_UAU.1.2

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)

5.1.3.4.1 FIA_UAU.7

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

5.1.3.5 Timing of identification (FIA_UID.1)

5.1.3.5.1 FIA_UID.1.1

The TSF shall allow no function on behalf of the user to be performed before the user is identified.

5.1.3.5.2 FIA_UID.1.2

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

5.1.3.6 User-subject binding (FIA_USB.1)[4]

5.1.3.6.1 FIA_USB.1

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

[4] The ST has an explicitly stated FIA_USB.1 requirement. Note the explicitly stated requirement does address interpretation I-415.

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

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

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

5.1.3.6.2 Note 2.1

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

a. The security attributes shall be a subset of those defined for the user.

5.1.3.6.3 Note 2.2

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

a. The effective user identity associated with a subject can be changed to another user's identity via a command, provided that successful authentication as the new user identity has been achieved;

b. When executing a file that has the set UID permission bit set, the effective user identity associated with the subject shall be changed to that of the owner of the file;

c. When executing a file that has the set GID permission bit set, the effective group identity associated with the subject shall be changed to that of the group attribute of the file.

5.1.4 SECURITY MANAGEMENT (FMT)

5.1.4.1 Management of object security attributes (FMT_MSA.1)

5.1.4.1.1 FMT_MSA.1.1

The TSF shall enforce the Discretionary Access Control Policy to restrict the ability to modify the access control attributes associated with a named object to the object owner or authorized administrator.

5.1.4.2 Static attribute initialization (FMT_MSA.3)

5.1.4.2.1 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.

5.1.4.2.2 FMT_MSA.3.2

The TSF shall allow the creator 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.1 (a))

5.1.4.3.1 FMT_MTD.1.1 (a)

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.1 (b))

5.1.4.4.1 FMT_MTD.1.1 (b)

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.1(c))

5.1.4.5.1 FMT_MTD.1.1(c)

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.1 (d))

5.1.4.6.1 FMT_MTD.1.1 (d)

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

5.1.4.6.2 FMT_MTD.1.1 (d)

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.1 (a))

5.1.4.7.1 FMT_REV.1.1 (a)

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

5.1.4.7.2 FMT_REV.1.2 (a)

The TSF shall enforce the rules:

a. The immediate revocation of security-relevant authorizations.

b. No other rules.

5.1.4.8 Revocation of object attributes (FMT_REV.1 (b))

5.1.4.8.1 FMT_REV.1.1 (b)

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.

5.1.4.8.2 FMT_REV.1.2 (b)

The TSF shall enforce the rules:

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

b. No other rules.

5.1.4.9 Security roles (FMT_SMR.1)

5.1.4.9.1 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, and

c. Users authorized to modify their own authentication data.

d. No other roles.

5.1.4.9.2 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)

5.1.5.1.1 FPT_AMT.1.1

The TSF shall run a suite of tests during initial start-up or 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 Non-bypassability of the TSP (FPT_RVM.1)

5.1.5.2.1 FPT_RVM.1.1

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

5.1.5.3 TSF domain separation (FPT_SEP.1)

5.1.5.3.1 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.

5.1.5.3.2 FPT_SEP.1.2

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

5.1.5.4 Reliable Time Stamp (FPT_STM.1)

5.1.5.4.1 FPT_STM.1.1

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

5.1.6 Strength of Function Requirement

The minimum strength of function level for the security functional requirements is SOF-medium.

5.2 TOE Security Assurance Requirements

The security assurance requirements for the TOE are the Evaluation Assurance Level (EAL) 3 components as specified in Part 3 of the Common Criteria. No operations are applied to the assurance components.

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

Table 3 EAL 3 Assurance Components

5.2.1 Configuration Management (ACM)

5.2.1.1 Authorization Controls (ACM_CAP.3)

5.2.1.1.1 ACM_CAP.3.1D

The developer shall provide a reference for the TOE.

5.2.1.1.2 ACM_CAP.3.2D

The developer shall use a CM system.

5.2.1.1.3 ACM_CAP.3.3D

The developer shall provide CM documentation.

5.2.1.1.4 Interpretation Note: The following element is added as a result of RI-003.

The configuration list shall uniquely identify all configuration items that comprise the TOE.

5.2.1.1.5 ACM_CAP.3.1C

The reference for the TOE shall be unique to each version of the TOE.

5.2.1.1.6 ACM_CAP.3.2C

The TOE shall be labeled with its reference.

5.2.1.1.7 ACM_CAP.3.3C

The CM documentation shall include a configuration list and CM plan.

5.2.1.1.8 ACM_CAP.3.4C

The configuration list shall describe the configuration items that comprise the TOE.

5.2.1.1.9 ACM_CAP.3.5C

The CM documentation shall describe the method used to uniquely identify the configuration items.

5.2.1.1.10 ACM_CAP.3.6C

The CM system shall uniquely identify all configuration items.

5.2.1.1.11 ACM_CAP.3.7C

The CM plan shall describe how the CM system is used.

5.2.1.1.12 ACM_CAP.3.8C

The evidence shall demonstrate that the CM system is operating in accordance with the CM plan.

5.2.1.1.13 ACM_CAP.3.9C

The CM documentation shall provide evidence that all configuration items have been and are being effectively maintained under the CM system.

5.2.1.1.14 ACM_CAP.3.10C

The CM system shall provide measures such that only authorized changes are made to the configuration items.

5.2.1.1.15 ACM_CAP.3.1E

The evaluator shall confirm that the information provided meets all the requirements for content and presentation of evidence

5.2.1.2 TOE CM Coverage (ACM_SCP.1)

5.2.1.2.1 ACM_SCP.1.1D

The developer shall provide ~~CM documentation~~ a list of configuration items for the TOE.[5]

5.2.1.2.2 ACM_SCP.1.1C

~~The CM documentation shall show that the CM system, as a minimum, tracks the following: the TOE implementation representation, design documentation, test documentation, user documentation, administrator documentation, and CM documentation.~~

The list of configuration items shall include the following: implementation representation and the evaluation evidence required by the assurance components in the ST.[6]

~~5.2.1.2.3 ACM_SCP.1.2C~~

~~The CM documentation shall describe how configuratio~~[7]~~n items are tracked by the CM system.~~

5.2.1.2.4 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)

5.2.2.1.1 ADO_DEL.1.1D

The developer shall document procedures for delivery of the TOE or parts of it to the user.

5.2.2.1.2 ADO_DEL.1.2D

The developer shall use the delivery procedures.

5.2.2.1.3 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.

5.2.2.1.4 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)

5.2.2.2.1 ADO_IGS.1.1D

The developer shall document procedures necessary for the secure installation, generation, and start-up of the TOE.

[5] Changed as a result of RI-004
[6] Changed as a result of RI-004 and RI-038.
[7] Changed as a result of RI-004.

5.2.2.2.2 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.[8]

5.2.2.2.3 ADO_IGS.1.1E

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.3 Development (ADV)

5.2.3.1 Informal Functional Specification (ADV_FSP.1)

5.2.3.1.1 ADV_FSP.1.1D

The developer shall provide a functional specification.

5.2.3.1.2 ADV_FSP.1.1C

The functional specification shall describe the TSF and its external interfaces using an informal style.

5.2.3.1.3 ADV_FSP.1.2C

The functional specification shall be internally consistent.

5.2.3.1.4 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.

5.2.3.1.5 ADV_FSP.1.4C

The functional specification shall completely represent the TSF.

5.2.3.1.6 ADV_FSP.1.1E

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.3.1.7 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)

5.2.3.2.1 ADV_HLD.2.1D

The developer shall provide the high-level design of the TSF.

5.2.3.2.2 ADV_HLD.2.1C

The presentation of the high-level design shall be informal.

[8] Changed as a result of RI-051

5.2.3.2.3 ADV_HLD.2.2C

The high-level design shall be internally consistent.

5.2.3.2.4 ADV_HLD.2.3C

The high-level design shall describe the structure of the TSF in terms of subsystems.

5.2.3.2.5 ADV_HLD.2.4C

The high-level design shall describe the security functionality provided by each subsystem of the TSF.

5.2.3.2.6 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.

5.2.3.2.7 ADV_HLD.2.6C

The high-level design shall identify all interfaces to the subsystems of the TSF.

5.2.3.2.8 ADV_HLD.2.7C

The high-level design shall identify which of the interfaces to the subsystems of the TSF are externally visible.

5.2.3.2.9 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.

5.2.3.2.10 ADV_HLD.2.9C

The high-level design shall describe the separation of the TOE into TSP-enforcing and other subsystems.

5.2.3.2.11 ADV_HLD.2.1E

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.3.2.12 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)

5.2.3.3.1 ADV_RCR.1.1D

The developer shall provide an analysis of correspondence between all adjacent pairs of TSF representations that are provided.

5.2.3.3.2 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.

5.2.3.3.3 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)

5.2.4.1.1 AGD_ADM.1.1D

The developer shall provide administrator guidance addressed to system administrative personnel.

5.2.4.1.2 AGD_ADM.1.1C

The administrator guidance shall describe the administrative functions and interfaces available to the administrator of the TOE.

5.2.4.1.3 AGD_ADM.1.2C

The administrator guidance shall describe how to administer the TOE in a secure manner.

5.2.4.1.4 AGD_ADM.1.3C

The administrator guidance shall contain warnings about functions and privileges that should be controlled in a secure processing environment.

5.2.4.1.5 AGD_ADM.1.4C

The administrator guidance shall describe all assumptions regarding user behavior that are relevant to secure operation of the TOE.

5.2.4.1.6 AGD_ADM.1.5C

The administrator guidance shall describe all security parameters under the control of the administrator, indicating secure values as appropriate.

5.2.4.1.7 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.

5.2.4.1.8 AGD_ADM.1.7C

The administrator guidance shall be consistent with all other documents supplied for evaluation.

5.2.4.1.9 AGD_ADM.1.8C

The administrator guidance shall describe all security requirements on the IT environment that are relevant to the administrator.

5.2.4.1.10 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)

5.2.4.2.1 AGD_USR.1.1D

The developer shall provide user guidance.

5.2.4.2.2 AGD_USR.1.1C

The user guidance shall describe the functions and interfaces available to the non-administrative users of the TOE.

5.2.4.2.3 AGD_USR.1.2C

The user guidance shall describe the use of user-accessible security functions provided by the TOE.

5.2.4.2.4 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.

5.2.4.2.5 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 behavior found in the statement of TOE security environment.

5.2.4.2.6 AGD_USR.1.5C

The user guidance shall be consistent with all other documentation supplied for evaluation.

5.2.4.2.7 AGD_USR.1.6C

The user guidance shall describe all security requirements on the IT environment that are relevant to the user.

5.2.4.2.8 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)

5.2.5.1.1 ALC_DVS.1.1D

The developer shall produce development security documentation.

5.2.5.1.2 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.

5.2.5.1.3 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.

5.2.5.1.4 ALC_DVS.1.1E

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.5.1.5 ALC_DVS.1.2E

The evaluator shall confirm that the security measures are being applied.

5.2.6 Security Testing (ATE)

5.2.6.1 Analysis of coverage (ATE_COV.2)

5.2.6.1.1 ATE_COV.2.1D

The developer shall provide an analysis of the test coverage.

5.2.6.1.2 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.

5.2.6.1.3 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.

5.2.6.1.4 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)

5.2.6.2.1 ATE_DPT.1.1D

The developer shall provide the analysis of the depth of testing.

5.2.6.2.2 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.

5.2.6.2.3 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)

5.2.6.3.1 ATE_FUN.1.1D

The developer shall test the TSF and document the results.

5.2.6.3.2 ATE_FUN.1.2D

The developer shall provide test documentation.

5.2.6.3.3 ATE_FUN.1.1C

The test documentation shall consist of test plans, test procedure descriptions, expected test results and actual test results.

5.2.6.3.4 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.

5.2.6.3.5 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.

5.2.6.3.6 ATE_FUN.1.4C

The expected test results shall show the anticipated outputs from a successful execution of the tests.

5.2.6.3.7 ATE_FUN.1.5C

The test results from the developer execution of the tests shall demonstrate that each tested security function behaved as specified.

5.2.6.3.8 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)

5.2.6.4.1 ATE_IND.2.1D

The developer shall provide the TOE for testing.

5.2.6.4.2 ATE_IND.2.1C

The TOE shall be suitable for testing.

5.2.6.4.3 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.

5.2.6.4.4 ATE_IND.2.1E

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.6.4.5 ATE_IND.2.2E

The evaluator shall test a subset of the TSF as appropriate to confirm that the TOE operates as specified.

5.2.6.4.6 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 (VLA)

5.2.7.1 Validation of analysis (AVA_MSU.1)

5.2.7.1.1 AVA_MSU.1.1D

The developer shall provide guidance documentation.

5.2.7.1.2 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.

5.2.7.1.3 AVA_MSU.1.2C

The guidance documentation shall be complete, clear, consistent and reasonable.

5.2.7.1.4 AVA_MSU.1.3C

The guidance documentation shall list all assumptions about the intended environment.

5.2.7.1.5 AVA_MSU.1.4C

The guidance documentation shall list all requirements for external security measures (including external procedural, physical and personnel controls).

5.2.7.1.6 AVA_MSU.1.1E

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.7.1.7 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.

5.2.7.1.8 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)

5.2.7.2.1 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.

5.2.7.2.2 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.

5.2.7.2.3 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.

5.2.7.2.4 AVA_SOF.1.1E

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.7.2.5 AVA_SOF.1.2E

The evaluator shall confirm that the strength claims are correct.

5.2.7.3 Independent vulnerability analysis (AVA_VLA.1)

5.2.7.3.1 AVA_VLA.1.1D

~~The developer shall perform and document an analysis of the TOE deliverables searching for obvious ways in which a user can violate the TSP.~~[9]

The developer shall perform a vulnerability analysis.

5.2.7.3.2 AVA_VLA.1.2D

~~The developer shall document the disposition of obvious vulnerabilities.~~[10]

The developer shall provide vulnerability analysis documentation.

[9] Changed as a result of RI-051
[10] Changed as a result of RI-051

5.2.7.3.3 AVA_VLA.1.1C

~~The documentation shall show, for all identified vulnerabilities, that the vulnerability cannot be exploited in the intended environment for the TOE.~~[11]

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.

The vulnerability analysis documentation shall describe the disposition of obvious vulnerabilities.

The vulnerability analysis documentation shall show, for all identified vulnerabilities, that the vulnerability cannot be exploited in the intended environment for the TOE.

5.2.7.3.4 AVA_VLA.1.1E

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.7.3.5 AVA_VLA.1.2E

The evaluator shall conduct penetration testing, building on the developer vulnerability analysis, to ensure obvious vulnerabilities have been addressed.

[11] Changed as a result of RI-051

6. TOE Summary Specification

This chapter describes the Mac OS X security functions and associated assurance measures.

6.1 TOE Security Functions

6.1.1 Audit

The audit security function provides for the generation and management of audit. Both of these areas are described in this section.

6.1.1.1 Audit Generation

Mac OS X maintains all audit records in one or more audit files. The audit files are managed by the audit daemon. The audit daemon is called by both kernel-model and user-mode processes to add audit records to the audit trail. When an audit record is created, the time stamp on the audit record is retrieved from the time daemon described in Section 6.1.5.4. The audit files are protected by discretionary access control (DAC) so that only administrators can access the audit trail.

Mac OS X provides a wide set of audit events to include those listed in Table 2 Auditable Events. Within each audit record, is the following information:

- Date
- Time
- Event type
- User Identifier (UID)
- Process ID (PID)
- Success or failure.

In addition to the information listed, for some audit records, additional information is provided as identified in Table 2 Auditable Events.

The administrator has two options when configuring how the audit trail will be managed. The administrator can specify a rotation scheme among the audit files. If the last in a series of audit files is full, Mac OS X will go back to the first audit file to start writing and overwrite any previously recorded data. The second option is once all audit files are filled, Mac OS X will halt. When the audit trails reaches 80% capacity, a message is written to the syslog to alert the administrator that the audit trail is reaching capacity.

6.1.1.2 Audit Management

The audit trail is protected so that only the administrator can access and manage the audit trail. The administrator has a number of graphical tools to manage the audit trail. The audit tools provide the following capabilities:

- Audit maintenance – the administrator can create, clear, and delete the audit trail.
- Audit selection – the administrator can select which audit events to record and can refine the selection based on user identity and success or failure of the audit event.
- Audit review – the administrator can review the contents of the audit trail. The audit trail viewing tool permits the administrator to search and sort the audit trail based on user identity.

The Audit security function satisfies the following security requirements:

- FAU_GEN.1 – Mac OS X minimally generates the events listed in the table referenced above and includes the date, time, event type, subject (PID), success or failure, as well as any additional content listed in the table above.
- FAU_GEN.2– Mac OS X records the responsible user in the contents of each audit record. The user identity is the target user for failed authentication attempts or the user authenticated for the session causing the event.
- FAU_SAR.1 – Using the audit management tools, the administrator can read and interrupt the audit trail.
- FAU_SAR.2 – The audit trail is protected by DAC so that only administrators can read the audit trail. No other users are given access to the audit trail.
- FAU_SAR.3 – Using the audit management tools, the administrator can search and sort the audit trail based on UID contained in the audit event.
- FAU_SEL.1 – Mac OS X includes the ability to filter audit records as they occur based on user identity and the event outcome.
- FAU_STG.1 – Mac OS X protects the audit records from deletion using DAC. The file permissions on the audit trail are such that only the administrator can access the audit files.
- FAU_STG.3 – If the audit trail exceeds 80% capacity, a warning message is written to the syslog so that the administrator is aware that audit trail is filling.
- FAU_STG.4 – Mac OS X has two options in the case the audit trail is full. First, the system can be configured to shut down if the audit trail fills. In this case, the administrator would need to clear disk space and reboot before the system will become operational again. In the second case, the administrator can choose to allow the oldest audit records to be overwritten if the audit trail fills.
- FMT_MTD.1 (a) – The graphical audit tools enable the administrator to perform a variety of audit management functions including creating, clearing, and deleting the audit log.
- FMT_MTD.1 (b) – Using the graphical audit tools, the administrator can view the set of audited events and can modify that list, if necessary.

6.1.2 Identification and Authentication

The identification and authentication security function provides logon features, user attribute management, and password and account management. Each is described in this section.

6.1.2.1 Logon Process

The logon process ensures that before a user uses any TSF-mediated functions, that user is identified and authenticated to Mac OS X. Users can either log onto the system locally or via a network service. When users log on at a local terminal, they are presented with a graphical interface requesting their user name and password. The user name is echoed back to the screen while the password is obscured from the user with dots. Authentication can take place using local passwords.

When a user requests some network services, the user must be authenticated. Network services either trust the identity of the client process or perform authentication on their own. The client process ssh passes an identity, which is trusted on the remote server, and is not required to re-authenticate unless the authentication fails; if authentication fails, the user is prompted for a password. Other network services/protocols in the evaluated configuration are unauthenticated. Those unauthenticated services/protocols are: TCP/IP, NFS, and DNS. The TCP/IP protocol relies on higher level services to perform authentication, NFS performs access checks based on the credentials of the requesting user, and DNS does not require authentication as it simply responds to name requests.

6.1.2.2 User Subject Binding

A process is used to associate users with subjects within the TOE. A process is an abstraction for a running program. A process's resources include a virtual address space, threads, and file descriptors. In Mac OS X, a process is based on one Mach task and one or more Mach threads. Every thread within a process has access to the address space and file descriptors. The security attributes associated with a process are:

- Real UID
- Saved UID
- Saved group identifier (GID)
- PID
- Effective UID
- Real GID
- Effective GID
- Group list
- Process umask.

There are three ways that a new subject can be created: a logon, a fork or exec, or a batch job is activated. When a logon occurs, a process is created on behalf of the requesting user and attributes are assigned at creation. When a running process issues a fork or exec, a copy of the calling process is created and it is assigned a new PID. Lastly, when a batch job is activated, a new process is created on behalf of the requesting user with the attributes of the requestor.

In most cases, the security attributes of a subject are derived from its creator; however, using a setuid or setgid program, or issuing the su command are exceptions. When executing a setuid or setgid program, the effective UID or GID, respectively, of the process will be changed. The new effective ID is taken from the file owner of the file being executed. Successful execution of the su program allows the subject to take on the targeted identity.

6.1.2.3 Account and Password Management

User account information is stored in the NetInfo database. The following information is stored for each user:

- User name
- User ID
- Password
- Groups.

The group information indicates if a user is part of the admin group. Being a member of the admin group makes a user an administrator, the only defined role on the system. There are two cases where users are permitted to perform certain role-like functions; those are users authorized by the DAC Policy to modify object security attributes, and users are authorized to modify their own authentication data. In these two special cases, users do not have to be the member of any group. All information is stored in files protected by DAC. Additionally, the password data is stored in encrypted format. Only the administrator has access to update the user account information. Changes to the authentication database take affect the next time a user logs on. If the administrator needs to immediately revoke a user's account attributes, the user must be forced to log off so the changes can take affect.

Passwords are initially assigned by the administrator and can be changed at any time by the administrator. Users are permitted to change their own passwords by supplying their current password and selecting a new password. The new password must have at least five characters. All characters are significant.

The Identification and Authentication security function satisfies the following security requirements:

- FIA_ATD.1 – The NetInfo database maintains users attributes including user name, user ID, password, and groups. The group field provides the indication of a user belonging to the administrator role.
- FIA_SOS.1 – Mac OS X provides a password mechanism to perform authentication. For each attempt to use the password mechanism, the probability that a random attempt will succeed is less than one in 1,000,000. For multiple attempts to use the password mechanism during a one-minute period, the probability that a random attempt during that minute will succeed is less than one in 100,000.
- FIA_UAU.1 – Mac OS X requires all users to authenticate before performing any TSF-mediated functions.
- FIA_UAU.7 – When a user enters a password, the characters are obscured with dot characters.
- FIA_UID.1 – Mac OS X requires all users to identify before performing any TSF-mediated functions.
- FIA_USB.1 – A process is the active entity that runs on behalf of a user. Associated with a process are user IDs, group IDs, umask, and process ID. These attributes are used to associate a user with auditable events and to enforce the DAC policy. The user attributes are assigned at subject creation time with the exception of running a setuid/setgid program or issuing a su command.
- FMT_MTD.1 (c) – Only the administrator is authorized to initialize and modify the user account database with the exception of password data.
- FMT_MTD.1 (d) – Only administrators can create password data. After the initial creation, users are permitted to modify their own passwords and administrators can modify any password.
- FMT_REV.1 (a) – Only administrators can revoke security attributes from a user. If immediate revocation is necessary, the administrator must force the user to log off.

6.1.3 User Data Protection

The user data protection security function provides discretionary access control (DAC) and residual information protection. This section describes both aspects of the security function.

6.1.3.1 Discretionary Access Control

The DAC mechanism is used to control access between subjects and named objects. DAC is a mechanism that controls access based on user and object identities. This section first identifies the attributes that are used when making DAC decisions and then describes the DAC policy for each type of object.

6.1.3.1.1 DAC Attributes

DAC decisions are based upon the attributes of subjects and objects. The DAC relevant attributes of a subject are:

- Real UID – UID established at logon
- Saved UID – UID saved when a setuid performed
- Saved GID - GID saved when a setgid performed
- PID – Process identifier
- Effective UID – Currently active UID
- Real GID - GID established at logon

- Effective GID - Currently active GID
- Group list – List of groups to which user belongs
- Process umask – Default permissions.

The DAC-specific attributes of the named objects depend upon the type of object. The following list identifies the object types followed by their associated DAC attributes.

- File System Objects (files, directories, named pipes, symbolic links, unnamed pipes)
    - Owner
    - Owning group
    - Permission bits
- Process Objects
    - Real UID
    - Effective UID
    - Saved UID
    - PID
    - Real GID
- Interprocess Communication (IPC) Mechanisms (shared memory segments, SysV/Posix semaphores, notifications, BSD locks)
    - Owner's UID
    - Owning GID
    - Creator's UID
    - Creator's GID
    - Permission bits

  Non-kernel objects (at jobs, crontab files, and print queue entries)
    - Owner

6.1.3.1.2 DAC Algorithm

DAC checks are made using either permissions bit checks or simple comparisons. This section describes how permission bit checking occurs. Simple comparisons are not described as they are straightforward and will be identified when they are used.

Permission bits divide permissions into three categories and users into three relative groups. The three categories of permissions are read, write, and execute. They are denoted as "r" for read, "w" for write, and "x" for execute. The three relative groups are the owner of the file, the owner's group, and every other user. When a check is performed against the permission bits, the owner bits are checked first, followed by the group bits, and concluded with the other bits. The first set of permissions matched is the permission granted.

6.1.3.1.3 DAC Policies

6.1.3.1.3.1 File System Objects

File system objects follow the DAC algorithm described in this section with one exception; administrators are always granted permission to file system objects. In order to access a file system object, a subject must pass the permissions bit check. Additionally, in order to read data from an object with a pathname, a

subject must have execute/search access to each directory in an object's pathname, and read access to the file. In order to write data into an object, a subject must have execute/search access to each directory in the pathname, and write access to the file. Directory objects have a special bit, called a *sticky bit*. When the sticky bit is set, only the owner of the directory may delete anything within the directory.

After a subject gains access to a file system object, the subject receives a file descriptor. In the file descriptor is a pointer to the file object and the permission granted. Whenever the subject makes future requests for the same object, the file descriptor is checked to determine that the subject is attempting to use only the permissions granted during the access check. In the case of unnamed pipes, no name exists in the namespace so all access occurs via file descriptors. An unnamed pipe is only accessible through the file descriptors given to the process that created it and the creating process' descendants. Because it has no name in the file system namespace, it has no pathname through which it can be accessed by unrelated processes

Access checks are performed slightly differently on symbolic links. The access checks are stored on the pathname stored in the symbolic link, not on the link itself.

6.1.3.1.3.2 Processes

A process can always read its own attributes. It can change its real UID and effective UID if it is root (UID=0) or if the requested value is equal to its real or saved UID. A process can also change its real GID, effective GID, and group list if it is root.

There is one instance where processes access one another. One process can always read the attributes of another process if it can name the target process using its PID.

6.1.3.1.3.3 IPC Mechanisms

Each IPC object includes a structure that contains: (a) the creator's effective UID and GID, (b) an owner effective UID and GID, and (c) a set of permission bits for creator and owner, creator's group and owner's group, and others. The owner UID and GID is initially set to the creator's UID and GID. Access is granted according to the permission bits.

6.1.3.1.3.4 Non-Kernel Objects

All non-kernel objects, at jobs, crontab files, and print queue entries, have their owner set to the user that submitted the request. All access is checked against the owner of the object. Only the owner or root may access any non-kernel object.

6.1.3.1.3.5 Default DAC

Only the owner or the administrator can modify the access control attributes associated with an object. The file system DAC permission bits are set to the value of the subject's umask at creation. The umask contains the initial permission settings and may be set as restrictively as the subject wants. The various IDs associated with objects are taken from the creating subject's attributes.

If a subject changes the permissions of an object, the changes take affect on the next access check against the object. So, if a subject has a file descriptor open for an object and attempts to use the file descriptor, the old permissions will remain in affect until the subject closes the object. If the subject closes the object and then attempts to re-open it, the new permissions would then be enforced.

6.1.3.2 Residual Data Protection

Mac OS X ensures that all previously allocated memory is cleared before is it allocated to a user process. File system objects are created with all fields initialized at creation time, overwriting the existing information. Additionally, an end of file marker prevents users from accessing data beyond the current file boundary. Other objects that use memory are cleared upon allocation. This includes process addresses space and execution context, as well as IPC memory spaces.

The User Data Protection security function satisfies the following security requirements:

- FDP_ACC.1 – The TSF enforces the DAC policy between all processes acting on the behalf of user and the following objects: files, directories, named pipes, symbolic links, unnamed pipes, shared memory segment, process, SysV/Posix semaphores, notifications, BSD locks, at jobs, crontab files, and print queue entries. The DAC policy is enforced on all operations between subjects and objects.
- FDP_ACF.1 – The TSF enforces the DAC policy to objects based on user identity and group membership(s) associated with a subject, and permission bits, object ownership, creator, and port rights associated with an object. See the descriptions in Section 6.1.3.1.3 for specific access policies.
- FDP_RIP.2 – The TSF ensures that previous information contents of resources used for new objects are not discernable in the new objects via clearing or overwriting of memory and tracking end of file pointers.
- Note 1 – All subjects are created with memory that has been cleared; thus, ensuring no residual information.
- FMT_MSA.1 – Only the owner and the administrator can change the access control attributes associated with an object.
- FMT_MSA.3 - The TSF provides restrictive default values for security attributes used to provide access control via the process's umask.
- FMT_REV.1 (b) – Object owners and the administrator can revoke the access rights associated with an object. The revocation takes place on the next attempt to open the object.

6.1.4 Security Management

Mac OS X supports security management by providing an administrator to manage the security functions. The administrator is authorized to perform all management functions including establishing and maintaining user accounts, modifying access rights, and managing the audit trail. The administrator account is realized via the use of a group. Members of the admin group are considered to be administrators.

The Security Management security function satisfies the following security requirement:

- FMT_SMR.1 – The TOE has an administrator role that can manage the security functions provided. Members of the admin group are considered to be administrators. Additionally, as described in the user data protection security function object owners can modify object security attributes. The identification and authentication security function states that users can modify their own authentication data.

6.1.5 TOE Protection Mechanisms

The TOE protection mechanisms security function provides abstract machine testing, reference mediation, domain separation, and reliable time stamps. Each function is described in this section.

6.1.5.1 Abstract Machine Testing

Since hardware and firmware are included in the TOE, both have tests to demonstrate their security features operate as claimed. These evaluation test suites include tests for memory protection and processor privileged instructions. The tests are made available to the administrator to run at the administrator's request. Apple also provides a set of hardware diagnostics to test that the hardware is operating correctly. The tests are run during system start-up and are called power-on self tests (POST).

6.1.5.2 Reference Mediation

The MAC OS X architecture is based on a kernel-mode architecture. The kernel executes in the kernel mode of the processor, which is the most privileged mode. Untrusted processes run in the user mode of the processor. The mechanism for entering the kernel mode also transfers control to the kernel, which then arbitrates any requests for service and access to resources based on the security policy and the file descriptor submitted by the user processes. When untrusted processes access system resources in user-mode, they do so through well–defined server interfaces so that the security policy is enforced on the untrusted processes.

6.1.5.3 Domain Separation

The software portion of the TOE uses several execution domains to protect itself from external interference and tampering. The kernel runs in the privileged mode (i.e., kernel-mode) provided by the evaluated processor architectures. A system call trap is a software interrupt operation that causes a context switch from user-mode into kernel-mode. In kernel-mode, a process can only execute the kernel defined code sequence that follows from a system call.

All TOE trusted servers (e.g., printing) and Administrator tools execute in a process, which is protected through the address space isolation mechanisms of the kernel. Each process is allocated a separate address space that is not shared unless specific access is granted.

6.1.5.4 Time

The system time is maintained and exported by the kernel using the time daemon. This system time is used in the audit trail to maintain an accurate accounting of when audit events occurred. Additionally the network time daemon and network time protocol (NTP) are used to synchronize clocks among networked computers.

The TOE Protection Mechanisms security function satisfies the following security requirements:

- FPT_AMT.1 – Mac OS X has two types of hardware diagnostic tests available for the administrator. The power-on self tests are provided with each machine and are run at system start-up. The tests developed during the evaluation test the security features of the hardware and can be run at the administrator's request.
- FPT_RVM.1 – All attempts to access system resources by untrusted, user-mode processes have to occur through one of two ways. First, requests can transfer of control to the kernel, which ensures access checks are made and enforced. Secondly, user processes can attempt to access other user-mode processes. In this case, access is through well-defined interfaces that implement the security policy.
- FPT_SEP.1 - The memory separation in the kernel ensures that processes can only access their own address space or address spaces explicitly shared; this protects trusted processes from untrusted processes. Untrusted processes are managed by the kernel and have separate address spaces and process contexts.
- FPT_STM.1 – The kernel reliably maintains the system time. Additionally, the network time daemon ensures synchronization among clocks on a network.

6.2 TOE Security Assurance Measures

The following assurance measures are applied to satisfy the Common Criteria EAL 3 assurance requirements:

- Configuration Management,
- Delivery and Operation,
- Development,

- Guidance Documents,
- Life Cycle Support,
- Security Testing, and,
- Vulnerability Assessment.

6.2.1 Configuration Management

The Configuration Management (CM) system applied by Apple ensures each product release is assigned a unique identifier. The CM system also identifies each hardware and software item that composes the TOE. The documentation and other programs managed by the CM system include: design documentation, test documentation, tests, user guide, administrator guide, and the configuration management plan. The CM plan is documented within the following document:

- Apple Configuration Management Plan, version 0.5, 16 December 2004

Assurance Requirements Satisfied: ACM_CAP.3 and ACM_SCP.1

6.2.2 Delivery and Operation

Apple has a set of Delivery and Operation documentation that describes the procedures for the delivery of the TOE. The documentation describes what is delivered with the TOE, instructions for installing and configuring the TOE, and warnings for the administrator to follow during installation. The Delivery and Operation documents are:

- Apple Delivery Procedures, version 0.5, 23 November 2004

Assurance Requirements Satisfied: ADO_DEL.1 and ADO_IGS.1

6.2.3 Development

Apple has a functional specification that describes the external interfaces of the TOE including the effects, exceptions, and error messages. There are also design documents that describe the security functions of the subsystems of the TOE. A correspondence exists that maps the high level design to the functional specification and the functional specification to the ST. The documents that meet the development assurance requirement:

Design Component Specifications:

- Administrative Tools, version 7, 27 December 2004
- Daemons, version 1.06, 29 November 2004
- Filesystem, version 2.06, 8 October 2004
- Hardware, version 3, 13 December 2004
- I/O Kit, version 5, 11 January 2005
- BSD/System V Inter-Process Communication, version 8, 24 November 2004
- Memory Management, version 4, 8 June 2004
- Microkernel, version 12, 2 September 2004
- Networking, version 6, 19 November 2004
- Process Management, version 3.6, 6 September 2004
- Security Framework, version 1.3, 12 May 2004

- User Interface, version 1.0, 29 April 2004
- SSH, version 0.05, 16 December 2004
- DNS, version 0.02, 15 March 2004

Assurance Requirements Satisfied: ADV_FSP.1, ADV_HLD.2, and ADV_RCR.

6.2.4 Guidance Documents

The Guidance Documents provided by Apple include both administrator and user manuals. The administrator manual describes the administrative functions and interfaces, provides guidance on how to administer the TOE securely, and contains warnings about functions and privileges that should be controlled. The user manual describes the functions and interfaces available to non-administrative users, describes the user-accessible security functions, and contains warnings to users about functions that should be controlled. The guidance documents are:

- Common Criteria Configuration and Administration, version 1.0, 17 December 2004

Assurance Requirements Satisfied: AGD_ADM.1 and AGD_USR.1

6.2.5 Life Cycle Support

The Life Cycle Support documentation describes how 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 it development environment are followed. The life cycle support document is

- Apple Life Cycle Manual, version 0.4, 20 December 2004

Assurance Requirements Satisfied: ALC_DVS.1

6.2.6 Security Testing

Apple maintains a security test suite consisting of test plans, procedures, expected results, and actual results. Apple has performed and documented an analysis that the test suite adequately tests the interfaces from both a coverage and depth perspective. A correspondence exists that maps the test suite to the functional specification. The test documents are:

- Apple Computer Mac OS X Version 10.3.6 Test Plan, version 0.6, 27 December 2004
- Component Test Specifications:
    - Admin Tools (GUI), version 2, 30 November 2004
    - Admin Tools (CLI) Test, version 2, 19 November 2004
    - Daemons Test, version 3, 13 November 2004
    - File System Test, version 7, 13 November 2004
    - NFS Filesystem Test, version 3, 12 November 2004
    - Hardware Test, version 1, 13 November 2004
    - I/O Kit Test, version 3, 22 November 2004
    - IPC Test, version 4, 12 November 2004
    - Memory Management Test, version 3, 11 November 2004
    - Microkernel Test, version 5, 19 November 2004
    - Networking Test, version 4, 12 November 2004
    - Process Management Test, version 6, 27 December 2004
    - Security Framework Test, version 3, 12 November 2004
    - User Networking Test, version 3, 13 November 2004

Assurance Requirements Satisfied: ATE_COV.2, ATE_DPT.1, ATE_FUN.1, and ATE_IND.2

6.2.7 Vulnerability Assessment

Apple has provided guidance documents that identify all possible modes of operation of the TOE, their consequences, and implications for maintaining secure operation. The guidance documents list all assumptions about intended usage and the TOE's environment.

The Strength of Function analysis performed on the password mechanism is provided in the following Apple document:

- Apple Computer Mac OS X Version 10.3.6 Strength of Function Analysis, version 2.0, 21 December 2004

As part of its design and testing process, Apple performs a vulnerability assessment. This analysis includes a search for obvious ways in which a user can violate the TSP. For all identified vulnerabilities, Apple provides an explanation why the vulnerability cannot be exploited in the intended environment for the TOE. The following documents the vulnerability analysis:

- Apple Computer Mac OS X and Mac OS X Server Vulnerability Assessment, version 0.4, 19 November 2004

Assurance Requirements Satisfied: AVA_MSU.1, AVA_SOF.1, and AVA_VLA.1

7. Protection Profile Claims

This section provides the PP conformance claims.

7.1 PP Identification

The TOE conforms to the Controlled Access Protection Profile, Version 1.d, October 8, 1999.

7.2 PP Tailoring

The following requirements from the Controlled Access Protection Profile were tailored in this Security Target:

- FAU_SAR.3 Selectable Audit Review
- FAU_SEL.1 Selective Audit
- FAU_STG.3 Action In Case Of Possible Audit Data Loss
- FAU_STG.4 Prevention of Audit Data Loss
- FDP_ACC.1 Discretionary Access Control
- FDP_ACF.1 Discretionary Access Control Functions
- FIA_ATD.1 Use Attribute Definition
- FIA_UAU.1 Timing of Authentication
- FIA_UID.1 Timing of Identification
- FIA_USB.1 User-Subject Binding
- FMT_MSA.1 Management of Object Security Attributes
- FMT_MSA.3 Static Attribute Initialization
- FMT_REV.1 (a) Revocation of User Attributes
- FMT_REV.1 (b) Revocation of Object Attributes
- FMT_SMR.1 Security Management Roles
- FPT_AMT.1 Abstract Machine Testing

7.3 PP Additions

No additions have been made to this Security Target. Two formatting changes have been made to this ST. The name of the FAU_STG.1 requirement has been corrected to conform to the CC. Secondly, a format has been defined and use for the iteration operation to more clearly distinguish requirements.

8. 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;
- Security Functional Requirement Dependencies;
- Explicitly Stated Requirements;
- TOE Summary Specification; and
- PP Claims.

8.1 Rationale For IT Security Objectives

The CAPP provides rationale for the security objectives demonstrating that security objectives are suitable to cover the intended environment. The rationale in the CAPP is valid for this ST as no new security objectives or environmental claims were added.

8.2 Rationale For Security Functional Requirements

The CAPP provides rationale for the security functional requirements demonstrating that security functional requirements are suitable to address the security objectives. The rationale in the CAPP is valid for this ST as no new security functional requirements or security objectives were added.

8.3 Security Assurance Requirements Rationale

The CAPP provides rationale for the security assurance requirements demonstrating that security assurance requirements are suitable for the intended environment. The rationale in the CAPP is valid for this ST as no new security assurance requirements or security objectives were added.

8.4 Requirement Dependency Rationale

The CAPP requirements have been evaluated and it has been determined that all dependencies have been satisfactorily addressed in the CAPP. Since this ST does not introduce any new requirements, no additional rationale is necessary.

8.5 Explicitly Stated Requirements Rationale

The ST does not contain any explicitly stated requirements. Therefore, this section is not applicable.

8.6 TOE Summary Specification Rationale

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 4 Security

Functions vs. Requirements Mapping demonstrates the relationship between security requirements and security functions. For a description of how the security functions satisfy the SFRs, see the mapping at the conclusion of each security function description in Section 6.1.

The only security mechanism that is realized by a probabilistic or permutational implementation is the password mechanism. By requiring passwords consist of five characters with at least two alphabetic characters and at least one numeric or special character, the claim exceeds the minimum strength of function requirement.

	AUDIT	I&A	USER DATA PROTECTION	MANAGEMENT	SELF PROTECT
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.2			X		
Note 1			X		
FIA_ATD.1		X			
FIA_SOS.1		X			
FIA_UAU.1		X			
FIA_UAU.7		X			
FIA_UID.1		X			
FIA_USB.1		X			
FMT_MSA.1			X		
FMT_MSA.3			X		
FMT_MTD.1(a)	X				
FMT_MTD.1 (b)	X				
FMT_MTD.1 (c)		X			
FMT_MTD.1 (d)		X			
FMT_REV.1 (a)		X			

	AUDIT	I&A	USER DATA PROTECTION	MANAGEMENT	SELF PROTECT
FMT_REV.1 (b)			X		
FMT_SMR.1				X	
FPT_AMT.1					X
FPT_RVM.1					X
FPT_SEP.1					X
FPT_STM.1					X

Table 4 Security Functions vs. Requirements Mapping

8.7 Rationale For PP Claims

The ST uses the claims made in the CAPP without modification. No environmental claims or security requirements have been added in this ST.

APPENDIX A List of Acronyms

CC	Common Criteria
CM	Configuration Management
DAC	Discretionary Access Control
DoD	Department of Defense
EAL	Evaluation Assurance Level
GID	Group Identifier
IPC	Interprocess Communication
IT	Information Technology
NKE	Network Kernel Extension
NTP	Network Time Protocol
PID	Process Identifier
POST	Power-on Self Test
PP	Protection Profile
SFR	Security Functional Requirement
ST	Security Target
TOE	Target of Evaluation
TSC	TSF Scope of Control
TSF	TOE Security Functions
UID	User Identifier