Red Hat® Certificate System (CAPP21) Security Target

Version 1.1
February 12, 2021

Prepared for:

Red Hat, Inc.

100 East Davie Street
Raleigh, NC 27601
USA

Prepared By:

Gossamer Laboratories
www.gossamersec.com

LIST OF TABLES

1. Security Target Introduction

This section identifies the Security Target (ST) and Target of Evaluation (TOE) identification, ST conventions, ST conformance claims, and the ST organization. The TOE is Red Hat Certificate System provided by Red Hat, Inc.. The TOE is being evaluated as a Certificate Authority.

The Security Target contains the following additional sections:

- Conformance Claims (Section 2)
- Security Objectives (Section 3)
- Extended Components Definition (Section 4)
- Security Requirements (Section 5)
- TOE Summary Specification (Section 6)

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 parenthetical number placed at the end of the component. For example FDP_ACC.1(1) and FDP_ACC.1(2) indicate that the ST includes two iterations of the FDP_ACC.1 requirement. Alternatively, iterations in this ST may use a forward slash followed by a descriptive text string, for example, FCS_CKM.1/TSF or FCS_COP.1/OE which denote that the SFR is met by the TSF or the OE (Operational Environment).
    - Assignment: allows the specification of an identified parameter. Assignments are indicated using bold and are surrounded by brackets (e.g., [**assignment**]). Note that an assignment within a selection would be identified in italics and with embedded bold brackets (e.g., [***[selected-assignment]***]).
    - Selection: allows the specification of one or more elements from a list. Selections are indicated using bold italics and are surrounded by brackets (e.g., [***selection***]).
    - Refinement: allows the addition of details. Refinements are indicated using bold, for additions, and strike-through, for deletions (e.g., "… **all** objects …" or "… ~~some~~ **big** things …").
- Other sections of the ST – Other sections of the ST use bolding to highlight text of special interest, such as captions.

1.1 Security Target Reference

ST Title – Red Hat Certificate System (CAPP21) Security Target

ST Version – Version 1.1

ST Date – February 12, 2021

1.2 TOE Reference

TOE Identification – Red Hat, Inc. Red Hat Certificate System 9.4 batch update 3

TOE Developer – Red Hat, Inc.

Evaluation Sponsor – Red Hat, Inc.

1.3 TOE Overview

The Target of Evaluation (TOE) is Red Hat Certificate System 9.4 batch update 3.

1.4 TOE Description

Red Hat® Certificate System (RHCS) is an enterprise software system that offers a scalable, secure framework to establish and maintain trusted identities and keep communications private.

Red Hat Certificate System provides certificate life-cycle management: issue, renew, suspend, revoke, archive/recover/manage single and dual-key X.509v3 certificates needed to handle strong authentication, single sign-on, and secure communications.

1.4.1 TOE Architecture

The RHCS TOE is an entity that issues and manages public-key certificates application written in Java, C++, C, and Perl using associated network (Network Security Services; NSS) and java (Java Security Services; JSS) security service libraries.

The RHCS is a software TOE that relies upon and incorporates different components in its Operational Environment. RHCS runs within Red Hat Enterprise Linux (RHEL 7.6), an operating system that protects the subsystems of the TOE with Security-Enhanced Linux (SELinux) policies and which provides secure network connections (using the TOE's Tomcat's HTTP/TLS to allow remote administration). The TOE integrates with a directory server in its Operational Environment, such as Red Hat Directory Server and to provide an internal data store. The underlying JSS and NSS in the RHEL operating system components of the TOE support the use of PKCS#11 hardware devices that perform standards-oriented cryptographic operations. All of the Operational Environment components along with the TOE components represent an RHCS system.
An RHCS system is composed of the following key components:

- Red Hat Certificate System (RHCS) - The Certificate System (CS) includes five configurable subsystems that work together to manage enterprise PKI deployments.
- Red Hat Enterprise Linux (RHEL) – The operating system (OS) in which RHCS and other components execute. The OS includes the software cryptographic module NSS, which supplies cryptographic functionality for TLS/HTTPS and which supplies cryptographic functionality for RHEL's package manager, RPM, which verifies update packages.
- Hardware Security Module (HSM) - –An HSM provides the FIPS-certified cryptographic services related to certificate management for the TOE. The HSM provides secure key storage for private keys as well as cryptographic services to allow secure use of stored keys (for example, using a stored key to sign a CRL). The tested configuration includes the Thales nShield Connect 6000+, model number NH2068. This HSM includes the nShield F3 6000+ for nShield Connect+ internal FIPS module (FIPS 2638). While the evaluation tested using this Thales HSM, any HSM that is at least FIPS 140-2 validated, provides PKCS#11 cryptographic services, hardware protection for keys and supports the required algorithms is considered equivalent.
- HTTP Engines (Tomcat (*for all subsystems: CA, KRA, OCSP Responder, TPS, and TKS*)) - The web engine provides the HTML-based UI (presentation), REST interface, and HTTP-based protocol handling. It does not perform authentication and authorization other than providing and/or enforcing TLS. It performs basic certificate validation and delegates all the application-specific authentication and authorization to CS via a callback mechanism.
- Internal Database (Red Hat Directory Server - RHDS 10.2) - The internal database stores information such as certificates, requests, officer/administrator/auditor information, and other information such as access control information. The CS communicates with the internal database securely through TLS client authentication.

The Certificate System is composed of the following subsystems running on one or more host systems. A subsystem can reside on its own host system or be combined with other subsystems on a single host system. While only the first three subsystems along with command line tools (installed as part of the RHCS packages) are responsible for enforcing the requirements of the CAPP21 Protection Profile requirements (i.e., they comprise the TSF), one may use the other subsystems in an evaluated configuration to provide additional functionality beyond the scope of the CAPP21. All RHCS subsystems share common frameworks such as authentication, authorization, and auditing, as well as utilize the same Operational Environment components.

- Certificate Authority (CA) - the subsystem that provides certificate management functionality for issuing, renewing, revoking, and publishing certificates and creating and publishing Certificate Revocation Lists (CRLs).
- Online Certificate Status Protocol (OCSP) Responder - a subsystem that provides OCSP responder services, based on stored CA's CRLs to distribute the load for certificate status verification.
- Key Recovery Authority (KRA) - a subsystem that provides private encryption key storage and retrieval. In a Token Management System, the KRA's HSM generates key pairs for the clients when server-side key generation option is turned on.
- Token Key Service (TKS) - manages one or more master keys required to set up secure channels from the tokens directly to the token processing system. The secure channels provided by TKS allows Global Platform compliant smart cards (tokens) to be identified with high level of confidence and subsequently communicate securely with the RHCS servers for operations such as certificate enrollments, renewals, server-side key generation requests, key archival and recovery, etc.
- Token Processing System (TPS) - one unique function of the TPS is to provide communication between Global Platform-compliant smart cards and the RHCS subsystems by means of APDU (Application Protocol Data Unit). It provides the registration authority functionality in the token management infrastructure and, with the assistance of the TKS, establishes secure channels between the smart cards and the back-end subsystems.

The CS subsystems (i.e., CA, KRA, OCSP Responder, TKS, and TPS) are highly integrated. OCSP and CA subsystems work together on CRL publishing and certificate verification. CA and KRA subsystems work together for key recovery and archival. Smart card tokens, processed through the Enterprise Security Client (ESC) user interface, are managed by the TPS. The TPS, however, works with at least two essential subsystems, a TKS to manage shared secrets between the tokens and the collective Token Management System (TMS) and a CA to process certificate enrollment operations. A TPS can also be configured to use a KRA for server-side key generation and key archival and recovery, with the assistance of TKS to deliver private keys securely to the tokens (smart cards).

The CA, KRA, OCSP Responder, TPS, and TKS are implemented in Java, utilize a Tomcat HTTP engine (see below), and share a common framework (also written in Java) for management, logging, authentication, access control, self-tests, and notifications framework.

The following architectural diagrams show the interactions between various CS configurations and various internal and external systems. Internally, the CS communicates with an internal database where certificate records, request records, and system user records are stored. The CS also accesses the cryptographic operations (directly or indirectly) via NSS. The RHEL components within the TOE Boundary includes, the HTTP engine which manages the presentation-level interaction between the CS and users including end-users, security officers, auditors, and administrators. The CS may optionally publish certificates to a corporate directory server.

In addition to the HTTP Engine and Internal Database, the CS also relies on access to processing capabilities, file storage, as well as hardware and software cryptographic modules provided by its Operational Environment.

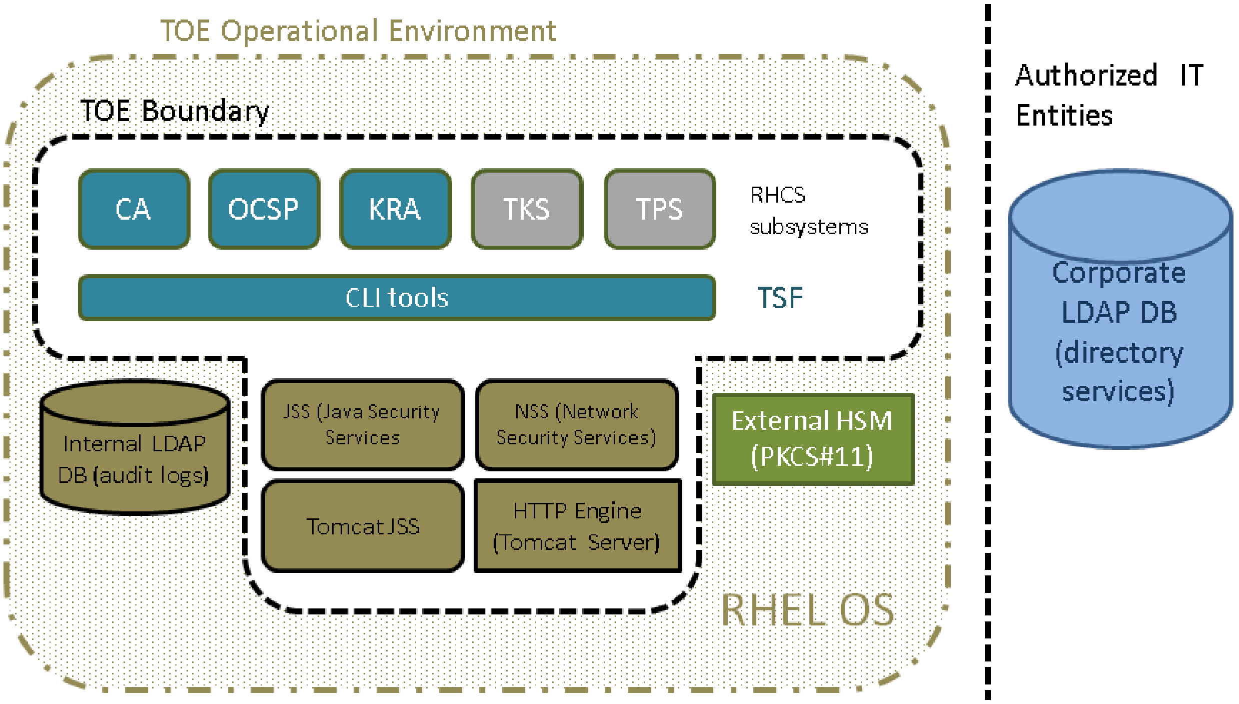
TOE Operational Environment
TOE Boundary
CA
OCSP
KRA
TKS
TPS
RHCS subsystems
CLI tools
TSF
Internal LDAP DB (audit logs)
JSS (Java Security Services
NSS (Network Security Services)
Tomcat JSS
HTTP Engine (Tomcat Server)
External HSM (PKCS#11)
RHEL OS
Authorized IT Entities
Corporate LDAP DB (directory services)

Figure 1-1 RHCS System Architecture

Figure 1-1 above depicts the architecture of the different parts of the TOE (including its RHCS [the upper portion] and RHEL [the lower, brown portion] components), the TOE's Operational Environment (OE), and external, authorized IT entities.

While a complete RHCS *system* includes all of the components indicated in Figure 1-1 the RHCS *TOE* includes the components within the TOE boundary. Specifically, the TOE's RHCS components consists of the CA, OCSP Responder, KRA, TKS, and TPS subsystems along with the command line utilities included with RHCS. The TOE's RHEL components include the NSS software cryptographic module and Tomcat server. The TSF portion of the TOE consists of the CA, OCSP, and KRA subsystems, while the TOE's OE includes other components within RHEL (the internal LDAP DB, watchdog daemon, and an HSM). The TOE OE's provides a Java GUI-based administration tool (called the "pkiconsole"). Administrators use this "console" for administrative tasks such as managing users and maintaining the different (CA, OCSP Manager, KRA, and TKS) subsystems and performing daily operational and managerial duties for those subsystems.

1.4.1.1 Physical Boundaries

As depicted in Figure 1-1, the TOE exists as a collection of application programs interacting with other components to implement its security functions. The TOE applications run within an IT environment based on RHEL (with configured SELinux policies) and including a Java runtime environment (with JSS/NSS libraries), a Tomcat HTTP Engine, and a directory server (e.g., Red Hat Directory Server) and watchdog daemon.

The TOE supports LDAP interfaces and also HTTP-based interfaces. The LDAP interfaces are used to connect to the internal LDAP Server (e.g., Red Hat Directory Server) used by RHCS exclusively as a private data store, and also to connect to a Corporate LDAP server for publishing purposes, if configured. The HTTP-based interfaces allow users, administrators, agents, and auditors to connect to RHCS to access its security functions and to manage RHCS.

Since the TOE is a collection of application programs, its logical and physical boundaries largely coincide. The TOE requires basic execution, data storage support, and network connectivity services from its IT environment. The external interfaces are limited to LDAP (over TLS), HTTP/TLS, and the use of command-line utility programs.

LDAP connections are supported only when initiated by RHCS. The HTTP/TLS interfaces are used to offer functions via service-oriented web pages to RHCS users, agents, auditors, and administrators. The command-line utility programs make use of these other interfaces, data files (e.g. for configuration or audit review), and in some cases do not interact with the rest of the TOE at all.

Note that many administrative functions (for the CA, KRA, OCSP, and TKS subsystems) are performed using a console application included with RHCS. This application interacts with the CS using HTTP/TLS, but instead of using XML/HTML it uses proprietary name/value pairs to better facilitate the administrator functions available. The TPS subsystem is managed via changes in configuration files (using RHEL OS tools) and through a web browser using HTTP/TLS. Some command-line tools are also available either utilizing the same name value pair mechanism or through the REST interface.

1.4.1.2 Logical Boundaries

This section summarizes the security functions provided by Red Hat Certificate System:

- Security audit
- Communication
- Cryptographic support
- User data protection
- Identification and authentication
- Security management
- Protection of the TSF
- TOE access
- Trusted path/channels

1.4.1.2.1 Security audit

The TOE generates logs for a range of security relevant events and relies upon its Operational Environment (OE) for generation of operating system events. The TOE provides secure storage of audit events and further provides separate audit storage for certificate related events. The TOE provides no administrator or auditor method for deletion or removal of events, and the TOE shuts down in the event of an error that prevents the TOE from creating new audit records.

1.4.1.2.2 Communication

The TOE provides proof of origin for issued certificates through CRLs and OCSP responses. The TOE also verifies certificate related messages using signed CMC requests and responses.

1.4.1.2.3 Cryptographic support

The TOE relies upon its OE for all cryptography and uses the OE-provided cryptography in support of certificate issuance and related CA operations, in support of HTTPS, TLSS, and TLSC operations.

1.4.1.2.4 User data protection

The TOE provides certificate profile functionality and certificate generation services conforming to IETF RFC 5280. The TOE provides certificate status information through CRLs and OCSP responses. The TOE clears sensitive data from buffers before releasing the buffers.

1.4.1.2.5 Identification and authentication

The TOE handles Certificate Management over CMS as both a client and server. The TOE performs certificate path validation in conformance with IETF RFC 5280.

1.4.1.2.6 Security management

The TOE provides all the interfaces necessary to manage the security functions identified throughout this Security Target as well as other functions commonly found in certificate authorities. The TOE provides its available functions to CA administrators, CA operations staff, Administrators/Officers, and Auditors.

1.4.1.2.7 Protection of the TSF

The RHCS TOE protects itself and also to rely on supporting protections from other components. At a high level, the TOE utilizes a separate and distinct hardware cryptographic engine for critical cryptographic operations; the TOE makes effective use of SELinux security mechanisms to protect itself and its underlying data and executables; the TOE command-line tools do not operate on or modify live TOE data, but rather use the documented security interfaces of the TOE to interact with the TOE; the TOE security functions are modular to isolate them from potential errors in other components; and the TOE interfaces are well-defined and restricted using a common certificate-based access control mechanism to distinguish among and limit the functions of administrator roles.

The TOE protects itself primarily using its identification & authentication and access control functions. With these functions, it ensures that users are properly authenticated and they are authorized to perform the functions made available by the TOE. Users that cannot be authenticated or that are not authorized will be denied access to applicable TOE functions.

The TOE relies on the components identified above for security and non-security functions. The primary security functions involve protecting the TOE as it is executing or at rest within its host, in facilitating secure inter-component communication, and providing FIPS-compliant cryptographic services.

The host operating system and Java implementation are relied upon to provide a distinct and separate execution environment for the TOE applications. In order to make effective use of the operating system, all RHCS components are packaged utilizing standard Red Hat package management (RPM). As such, whenever the TOE components are installed, they are stored with "root" user and group ownership and utilize standard Linux directory, file, and executable UNIX permissions. When an RHCS TOE instance is generated from these installed components, a "pkiuser" user and group identifier is used for ownership of *most* portions of the installed instances. The notable exceptions are (1) that an instance's start/stop script is ONLY granted "root" ownership with read/write/execute permission available only to root and (2) that the signed audit log files contained under the signedAudit directory contain a group privilege of "pkiaudit" to allow separation of roles between auditors and administrators. Files owned by "pkiuser" containing potentially sensitive information (e. g., log files, configuration files such as CS.cfg, and NSS security database files) contain no privileges for "other" users (e.g., file permissions of 00660 or 00600). Also, the entire contents of each PKI instance's signed audit directory are not accessible to "other" users. In practice, access to the "root" account is limited to administrators and the "pkiuser" account is configured so that it is not used by any human user, but rather is used by TOE components.

While previous versions of the TOE operated in an unconfined SELinux domain, a SELinux policy was created specifically to enhance the protection of RHCS. This policy includes the following characteristics:

1. The files and directories delivered for each of the subsystems are labeled with specific SELinux contexts (*pki_ca_exec_t*, *pki_ca_var_lib_t*, *pki_ca_var_log_t*, etc. for a CA for example).
2. The ports used by each subsystem are labeled with specific SELinux contexts (*pki_ca_port*, *pki_tps_port,* etc.).
3. The CS subsystem processes are also constrained to run within specific SELinux domains (*pki_ca_t*, *pki_ra_t*, *pki_ocsp_t*, etc.). When processes are started, they start in the *unconfined_t* domain, but transition into their assigned domain.
4. Each SELinux domain has rules written to specify the actions that are authorized for the domain. As an example, the *pki_ca_t* domain has rules written to allow write-access files with context *pki_ca_var_log_t*. Moreover, it has rules to allow processes running within the domain to connect to ports of type *pki_ca_port* (as well as others).
5. All accesses not specified in the policy are denied.

Ultimately, the operating system with SELinux extensions is configured to protect the TOE and its stored data using the core access control mechanisms and SELinux domain protection mechanisms.

The TOE also relies on its security providers (JSS/NSS) and web engines primarily to facilitate secure (TLS/HTTPS) communications between TOE components and also with TOE clients. While the TOE can support a number of cipher suites with RSA and ECC key exchange, limiting TLS ciphers to FIPS compliant algorithms is encouraged.

Finally, the TOE depends on a FIPS validated HSM to provide the underlying cryptographic support necessary to allow the TOE to securely act as a certificate authority (signing/issuing certificates and revocation information [CRL and OCSP]). The TOE accesses the HSM via a corresponding library installed on the host operating system. The HSM stores critical keys so that they are not externally accessible. It provides access to its embedded keys in order to generate new keys, encrypt/decrypt data, produce signatures, etc. In practice, the TOE is the sole user or client of the HSM attached directly to its host operating system.

1.4.1.2.8 TOE access

The TOE offers an administrator configurable timeout after which to lock remote interactive sessions as well as allowing remote users to terminate their interactive session. The TOE also has the capability to display an advisory message (banner) when users access the TOE for use.

1.4.1.2.9 Trusted path/channels

The TOE protects interactive communication with administrators on the HTTPS (WebUI) interface, the set of TLS protected command line tools, and the pkiconsole application that utilizes HTTPS protected REST API interfaces. In each case, both integrity and disclosure protection are ensured. If the negotiation of an encrypted session fails or if the user does not have authorization for remote administration, the attempted connection will not be established.

The TOE protects communication with network peers, such as a directory services, using TLS connections to prevent unintended disclosure or modification of data.

1.4.2 TOE Documentation

Red Hat offers documents that describe the installation and use of Red Hat Certificate System. The following list of documents was examined as part of the evaluation.

- Red Hat Certificate System 9.4 Planning, Installation, and Deployment Guide (Common Criteria Edition), Revision 9.4-1 - February 2021
- Red Hat Certificate System 9.4 Administration Guide (Common Criteria Edition), Revision 9.4-1 - February 2021

2. Conformance Claims

This TOE is conformant to the following CC specifications:

- Common Criteria for Information Technology Security Evaluation Part 2: Security functional components, Version 3.1, Revision 5, April 2017.
    - Part 2 Extended
- Common Criteria for Information Technology Security Evaluation Part 3: Security assurance components, Version 3.1 Revision 5, April 2017.
    - Part 3 Conformant
- Package Claims:
    - Protection Profile for Certification Authorities, Version 2.1, 01 December 2017 (CAPP21)
- Technical Decisions: TD0276, TD0278, TD0286, TD0287, TD0294, TD0328, TD0348, TD0353, TD0375

2.1 Conformance Rationale

The ST conforms to the CAPP21. As explained previously, the security problem definition, security objectives, and security requirements have been drawn from the PP.

3. Security Objectives

The Security Problem Definition may be found in the CAPP21 and this section reproduces only the corresponding Security Objectives for operational environment for reader convenience. The CAPP21 offers additional information about the identified security objectives, but that has not been reproduced here and the CAPP21 should be consulted if there is interest in that material.

In general, the CAPP21 has defined Security Objectives appropriate for certificate authorities and as such are applicable to the Red Hat Certificate System TOE.

3.1 Security Objectives for the Operational Environment

OE.AUDIT_GENERATION The Operational Environment provides a mechanism for the generation of portions of the audit data.

OE.AUDIT_RETENTION The Operational Environment provides mechanisms for retention of audit records for both normal and extended retention periods.

OE.AUDIT_REVIEW The Operational Environment provides a mechanism for the review of specified audit data.

OE.AUDIT_STORAGE The Operational Environment provides a mechanism for the storage of specified audit data.

OE.CERT_REPOSITORY The Operational Environment provides a certificate repository for storage of certificates (and optionally CRLs) issued by the TSF.

OE.CERT_REPOSITORY_SEARCH The Operational Environment provides the ability to search a certificate repository for specific certificate fields in certificates issued by the TSF and return the certificate and an identifier for the certificate that can be used to search the audit trail for events related to that certificate.

OE.CRYPTOGRAPHY The Operational Environment provides cryptographic services that can be invoked by the TSF in order to perform security functionality.

OE.KEY_ARCHIVAL The Operational Environment provides the ability to use split knowledge procedures to enforce two-party control to export keys necessary to resume CA functionality if the TSF should fail.

OE.NO_GENERAL_PURPOSE There are no general-purpose computing capabilities (e.g., compilers or user applications) available on the TOE, other than those services necessary for the operation, administration and support of the TOE.

OE.PHYSICAL Physical security, commensurate with the value of the TOE and the data it contains, is provided by the environment.

OE.PUBLIC_KEY_PROTECTION The Operational Environment provides protection for specified public keys associated with CA functions.

OE.SESSION_PROTECTION_LOCAL The Operational Environment provides the ability to lock or terminate local administrative sessions.

OE.SESSION_PROTECTION_REMOTE The Operational Environment provides the ability to lock or terminate remote administrative sessions.

OE.TOE_ADMINISTRATION The Operational Environment provides specified management capabilities required for the overall operation of a Certificate Authority, and the ability to restrict access to a subset of the capabilities as specified in the ST.

OE.TRUSTED_ADMIN The administrator of the TOE is not careless, willfully negligent or hostile, and administers the software within compliance of the applied enterprise security policy.

OE.TRUSTED_PLATFORM The operating system on which the TOE has been installed is securely configured, regularly patched, and not subject to unauthorized access.

4. Extended Components Definition

All of the extended requirements in this ST have been drawn from the CAPP21. The CAPP21 defines the following extended requirements and since they are not redefined in this ST the CAPP21 should be consulted for more information in regard to those CC extensions.

Extended SFRs:

- FAU_ADP_EXT.1: Audit Dependencies
- FAU_GCR_EXT.1: Generation of Certificate Repository
- FAU_STG_EXT.1: External Audit Trail Storage
- FCO_NRO_EXT.2: Certificate-Based Proof of Origin
- FCO_NRR_EXT.2: Certificate-Based Proof of Receipt
- FCS_CDP_EXT.1: Cryptographic Dependencies
- FCS_HTTPS_EXT.1: Extended: HTTPS Protocol
- FCS_STG_EXT.1: Cryptographic Key Storage
- FCS_TLSC_EXT.2: TLS Client Protocol with Mutual Authentication
- FCS_TLSS_EXT.2: TLS Server Protocol with Mutual Authentication
- FDP_CER_EXT.1: Certificate Profiles
- FDP_CER_EXT.2: Certificate Request Matching
- FDP_CER_EXT.3: Certificate Issuance Approval
- FDP_CRL_EXT.1: Certificate Revocation List Validation
- FDP_CSI_EXT.1: Certificate Status Information
- FDP_OCSPG_EXT.1: OCSP Basic Response Generation
- FIA_CMC_EXT.1: Certificate Management over CMS (CMC)
- FIA_UAU_EXT.1: Authentication Mechanism
- FIA_UIA_EXT.1: User Identification and Authentication
- FIA_X509_EXT.1: Certificate Validation
- FIA_X509_EXT.2: Certificate-Based Authentication
- FPT_KST_EXT.1: No Plaintext Key Export
- FPT_KST_EXT.2: TSF Key Protection
- FPT_SKP_EXT.1: Protection of Keys
- FPT_SKY_EXT.1(2)/OTH: Split Knowledge Procedures - OTH
- FPT_TUD_EXT.1: Trusted Update

5. Security Requirements

This section defines the Security Functional Requirements (SFRs) and Security Assurance Requirements (SARs) that serve to represent the security functional claims for the Target of Evaluation (TOE) and to scope the evaluation effort.

The SFRs have all been drawn from the CAPP21. The refinements and operations already performed in the CAPP21 are not identified (e.g., highlighted) here, rather the requirements have been copied from the CAPP21 and any residual operations have been completed herein. Of particular note, the CAPP21 made a number of refinements and completed some of the SFR operations defined in the Common Criteria (CC) and that PP should be consulted to identify those changes if necessary.

The SARs are also drawn from the CAPP21 which includes all the SARs for EAL 1. However, the SARs are effectively refined since requirement-specific 'Assurance Activities' are defined in the CAPP21 that serve to ensure corresponding evaluations will yield more practical and consistent assurance than the EAL 1 assurance requirements alone. The CAPP21 should be consulted for the assurance activity definitions.

5.1 TOE Security Functional Requirements

The following table identifies the SFRs that are satisfied by Red Hat Certificate System TOE.

Requirement Class	Requirement Component
FAU: Security audit	FAU_ADP_EXT.1: Audit Dependencies
	FAU_GCR_EXT.1: Generation of Certificate Repository
	FAU_GEN.1: Audit Data Generation
	FAU_GEN.2: User Identity Association
	FAU_SAR.1: Audit review
	FAU_SAR.3: Selectable Audit Review
	FAU_STG.1(1): Protected Audit Trail Storage (Normal)
	FAU_STG.4: Prevention of Audit Data Loss
	FAU_STG_EXT.1: External Audit Trail Storage
FCO: Communication	FCO_NRO_EXT.2: Certificate-Based Proof of Origin
	FCO_NRR_EXT.2: Certificate-Based Proof of Receipt
FCS: Cryptographic support	FCS_CDP_EXT.1: Cryptographic Dependencies
	FCS_CKM.1: Cryptographic Key Generation
	FCS_CKM.2: Cryptographic Key Establishment
	FCS_CKM_EXT.4: Cryptographic Key Destruction
	FCS_COP.1(1): Cryptographic Operation (AES Encryption/Decryption)
	FCS_COP.1(2): Cryptographic Operation (Cryptographic Signature)
	FCS_COP.1(3): Cryptographic Operation (Cryptographic Hashing)
	FCS_COP.1(4): Cryptographic Operation (Keyed-Hash Message Authentication)
	FCS_HTTPS_EXT.1: HTTPS Protocol
	FCS_RBG_EXT.1: Cryptographic Random Bit Generation
	FCS_STG_EXT.1: Cryptographic Key Storage
	FCS_TLSC_EXT.2: TLS Client Protocol with Mutual Authentication
	FCS_TLSS_EXT.2: TLS Server Protocol with Mutual Authentication
FDP: User data protection	FDP_CER_EXT.1: Certificate Profiles
	FDP_CER_EXT.2: Certificate Request Matching
	FDP_CER_EXT.3: Certificate Issuance Approval
	FDP_CRL_EXT.1: Certificate Revocation List Validation
	FDP_CSI_EXT.1: Certificate Status Information

	FDP_ITT.1: Basic Internal Transfer Protection
	FDP_OCSPG_EXT.1: OCSP Basic Response Generation
	FDP_RIP.1: Subset Residual Information Protection
FIA: Identification and authentication	FIA_CMCS_EXT.1: Certificate Management over CMS (CMC) Server
	FIA_ENR_EXT.1: Certificate Enrollment
	FIA_UAU_EXT.1: Authentication Mechanism
	FIA_UIA_EXT.1: User Identification and Authentication
	FIA_X509_EXT.1: Certificate Validation
	FIA_X509_EXT.2: Certificate-Based Authentication
	FIA_X509_EXT.3: X509 Certificate Request
FMT: Security management	FMT_MOF.1(1): Management of Security Functions Behavior (Administrator Functions)
	FMT_MOF.1(2): Management of Security Functions Behavior (CA/RA Functions)
	FMT_MOF.1(3): Management of Security Functions Behavior (CA Operations Functions)
	FMT_MOF.1(4): Management of Security Functions Behavior (Admin/Officer Functions)
	FMT_MOF.1(5): Management of Security Functions Behavior (Auditor Functions)
	FMT_MTD.1: Management of TSF Data
	FMT_SMF.1: Specification of Management Functions
	FMT_SMR.2: Restrictions on Security Roles
FPT: Protection of the TSF	FPT_FLS.1: Failure with Preservation of Secure State
	FPT_ITT.1: Basic Internal TSF Data Transfer Protection
	FPT_KST_EXT.1: No Plaintext Key Export
	FPT_KST_EXT.2: TSF Key Protection
	FPT_RCV.1: Manual Trusted Recovery
	FPT_SKP_EXT.1: Protection of Keys
	FPT_SKY_EXT.1(2)/OTH: Split Knowledge Procedures
	FPT_STM.1: Reliable Time Stamps
	FPT_TUD_EXT.1: Trusted Update
FTA: TOE access	FTA_SSL.3: TSF-Initiated Termination
	FTA_SSL.4: User-Initiated Termination
	FTA_TAB.1: Default TOE Access Banners
FTP: Trusted path/channels	FTP_ITC.1: Inter-TSF Trusted Channel
	FTP_TRP.1: Trusted Path

Table 5-1 TOE Security Functional Components

5.1.1 Security audit (FAU)

5.1.1.1 Audit Dependencies (FAU_ADP_EXT.1)

FAU_ADP_EXT.1.1

The TSF shall implement audit functionality and [***interface with auditing function(s) in the Operational Environment***] in order to perform audit operations on the following audit data: [***the auditable events denoted as persistent in Table 5-2 - Auditable Events***].

5.1.1.2 Generation of Certificate Repository (FAU_GCR_EXT.1)

FAU_GCR_EXT.1.1

The TSF shall [***invoke the Operational Environment to store***] certificates and [***CRLs***] issued by the TSF.

5.1.1.3 Audit Data Generation (FAU_GEN.1)

FAU_GEN.1.1

The TSF shall be able to generate and [***invoke the Operational Environment to generate***] an audit record of the following auditable events:

a) Start-up of the TSF audit functions;
b) All auditable events for the not specified level of audit; and
c) All administrative actions invoked through the TSF interface;
d) Specifically defined auditable events listed in **Table 5-2 - Auditable Events**.

Requirement	**Auditable Events**	**Additional Content**	**Retained Nor/Ext**	**Responsible TSF or OE Component**
FAU_ADP_EXT.1				
FAU_GEN.1				
FAU_GEN.2				
FAU_SAR.1				
FAU_SAR.3				
FAU_GCR_EXT.1				
FAU_STG.1(1)	Any attempt to delete the audit log.			OE - RHEL
FCO_NRO_EXT.2				
FCO_NRR_EXT.2				
FCS_CDP_EXT.1				
FCS_CKM.1	All occurrences of non-ephemeral and [no other] key generation for TOE related functions.	Success: public key generated	Extended	TSF – CA
FCS_CKM.2	All occurrences of non-ephemeral and [no other] key generation for TOE related functions.	Success: public key generated	Extended	N/A
FCS_CKM_EXT.4				
FCS_COP.1(1)				
FCS_COP.1(2)	All occurrences of signature generation using a CA signing key. Failure in signature generation.	Name/identifier of object being signed Identifier of key used for signing.	Extended	TSF - CA
FCS_COP.1(3)				
FCS_COP.1(4)				
FCS_COP.1(5)	None	None	N/A	N/A
FCS_HTTPS_EXT.1	Failure to establish a HTTPS session. Establishment/Termination of a HTTPS session.	Reason for failure. Non-TOE endpoint of connection (IP address) for both successes and failures.	Normal	TSF – each subsystem
FCS_RBG_EXT.1				
FCS_STG_EXT.1				
FCS_TLSC_EXT.2	Failure to establish a TLS Session. Establishment/Termination of a TLS session.	Reason for failure.	Normal	TSF – each subsystem
FCS_TLSS_EXT.2	Failure to establish a TLS Session. Establishment/Termination of a TLS session.	Reason for failure.	Normal	TSF – each subsystem
FDP_CER_EXT.1	Certificate generation.	Success: [***certificate object identifier***].	Extended	TSF - CA
FDP_CER_EXT.2	Linking of certificate to certificate request.	Success: [***certificate value, certificate object identifier***], [***link to certificate request object identifier***].	Extended	TSF - CA

		Failure: Reason for failure, [***link to certificate request object identifier***].		
FDP_CER_EXT.3	Failed certificate approvals.	Reason for failure. [***link to certificate request object identifier***].	Extended	TSF - CA
FDP_CRL_EXT.1	Failure to generate CRL.		Extended	TSF - CA
FDP_CSI_EXT.1				
FDP_ITT.1				
FDP_OCSPG_EXT.1	Failure to generate certificate status information.		Extended	TSF - OCSP
FDP_RIP.1				
FIA_AFL.1	The reaching of the threshold for the unsuccessful authentication attempts. The action taken. The reenablement of disabled nonadministrative accounts.		Normal	
FIA_CMCS_EXT.1	CMC requests (generated or received) containing certificate requests or revocation requests. CMC responses issued.	Identifiers for all entities authenticating the request, including the entity providing client authentication for the EST transport (if any). The submitted request. Any signed response.	Extended	TSF - CA
FIA_ENR_EXT.1				
FIA_UAU_EXT.1				
FIA_UIA_EXT.1	All use of the identification and authentication mechanism.	Provided user identity. Origin of the attempt (e.g., IP address).	Normal	TSF – each subsystem
FIA_X509_EXT.1	Failed certificate validations.		Normal	TSF – each subsystem
FIA_X509_EXT.2	Failed authentications.		Normal	TSF – each subsystem
FIA_X509_EXT.3				
FMT_MOF.1(1)				
FMT_MOF.1(2)				
FMT_MOF.1(3)				
FMT_MOF.1(4)				
FMT_MOF.1(5)				
FMT_MTD.1				
FMT_SMF.1				
FMT_SMR.2	Modifications to the group of users that are part of a role.	Modifications to the group of users that are part of a role.	Extended	TSF – each subsystem
FPT_FLS.1	Invocation of failures under this requirement	Indication that the TSF has failed with the type of failure that occurred.	Normal	TSF – each subsystem
FPT_ITT.1				
FPT_KST_EXT.1				
FPT_KST_EXT.2	All unauthorized attempts to use TOE secret and private keys.	Identifier of user or process that attempted access.	Normal	TSF – each subsystem
FPT_RCV.1 (TD0286 applied)	The fact that a failure or service discontinuity occurred; resumption of the regular operation.	The type of failure or service discontinuity.	Extended	TSF – each subsystem
FPT_SKP_EXT.1				
FPT_SKY_EXT.1(2) /OTH				
FPT_STM.1	Changes to the time.	The old and new values for the time.	Normal	OE - RHEL
FPT_TUD_EXT.1	Initiation of update.	Version number	Extended	OE - RHEL
FTA_SSL.3	The termination of a remote session by the session locking mechanism.		Normal	TSF – each subsystem
FTA_SSL.4	The termination of an interactive session.		Normal	TSF – each subsystem
FTA_TAB.1				
FTP_ITC.1	Initiation of the trusted channel.	Identification of the	Normal	TSF – each subsystem

	Termination of the trusted channel. Failure of the trusted channel functions.	initiator and target of failed trusted channels establishment attempt.		
FTP_TRP.1	Initiation of the trusted channel. Termination of the trusted channel. Failure of the trusted channel functions.	Identification of the initiator and target of failed trusted channels establishment attempt.	Normal	TSF – each subsystem

Table 5-2 - Auditable Events

FAU_GEN.1.2

Refinement: The TSF shall [***include, invoke the Operational Environment to include***] 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) For each audit event type, based on the auditable event definitions of the functional components included in the PP/ST, information specified in column three of **Table 5-2 - Auditable Events**.

5.1.1.4 User Identity Association (FAU_GEN.2)

FAU_GEN.2.1

For audit events resulting from actions of identified users, the TSF shall be able to [***associate, invoke the Operational Environment to associate***] each auditable event with the identity of the user that caused the event.

5.1.1.5 Audit review (FAU_SAR.1)

FAU_SAR.1.1

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

FAU_SAR.1.2

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

5.1.1.6 Selectable Audit Review (FAU_SAR.3)

FAU_SAR.3.1

The TSF shall provide the ability to apply searches of audit data based on [***certificate serial number***] associated with the event.

5.1.1.7 Protected Audit Trail Storage (Normal) (FAU_STG.1(1))

FAU_STG.1(1).1

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

FAU_STG.1(1).2

The TSF shall be able to prevent unauthorized modifications to the stored audit records in the audit trail.

5.1.1.8 Prevention of Audit Data Loss (FAU_STG.4)

FAU_STG.4.1

The TSF shall prevent audited events, except those taken by the Auditor and [***shutdown***] if the audit trail cannot be written to.

5.1.1.9 External Audit Trail Storage (FAU_STG_EXT.1)

FAU_STG_EXT.1.1

The TSF shall maintain availability and integrity of audit data by storing it [***locally on the TOE platform***].

5.1.2 Communication (FCO)

5.1.2.1 Certificate-Based Proof of Origin (FCO_NRO_EXT.2)

FCO_NRO_EXT.2.1

The TSF shall provide proof of origin for certificates it issues in accordance with the digital signature requirements using a mechanism in accordance with RFC 5280 and FCS_COP.1(2).

FCO_NRO_EXT.2.2

The TSF shall provide proof of origin for certificate status information it issues in accordance with the digital signature requirements in [***CRLs (RFC 5280), OCSP (RFC 6960)***] and FCS_COP.1(2).

FCO_NRO_EXT.2.3

The TSF shall require and verify proof of origin for certificate requests it receives [***CMC using mechanisms in accordance with FIA_CMC_EXT.1***].

FCO_NRO_EXT.2.4

The TSF shall require and verify proof of origin for public keys contained in certificate requests it receives via [***proof-of-possession mechanisms in CMC using mechanisms in accordance with FIA_CMC_EXT.1***].

FCO_NRO_EXT.2.5

The TSF shall [

- ***require and verify proof of origin for revocation requests it receives via [CMC using mechanisms in accordance with FIA_CMCS_EXT.1]*]**.

5.1.2.2 Certificate-Based Proof of Receipt (FCO_NRR_EXT.2)

FCO_NRR_EXT.2.1

The TSF shall provide proof of receipt for [***CMC***] by providing signed responses using mechanisms in accordance with [***FIA_CMCS_EXT.1***].

5.1.3 Cryptographic support (FCS)

5.1.3.1 Cryptographic Dependencies (FCS_CDP_EXT.1/TSF)

FCS_CDP_EXT.1.1(TSF)

The TSF shall [***implement cryptographic functionality***] in order to perform [

- ***FCS_CKM.1/TSF,***
- ***FCS_CKM.2/TSF,***
- ***FCS_CKM_EXT.4/TSF,***
- ***FCS_COP.1(1/TSF)(AES Encryption/Decryption),***
- ***FCS_COP.1(2/TSF)(Cryptographic Signature),***
- ***FCS_COP.1(3/TSF)(Cryptographic Hashing),***
- ***FCS_COP.1(4/TSF)(Keyed-Hash Message Authentication),***
- ***FCS_RBG_EXT.1***] cryptographic operations.

5.1.3.2 Cryptographic Dependencies (FCS_CDP_EXT.1/OE)

FCS_CDP_EXT.1.1(OE)

The TSF shall [***invoke interfaces provided by the Operational Environment***] in order to perform [

- ***FCS_CKM.1/OE,***
- ***FCS_COP.1(1/OE)(AES Encryption/Decryption),***
- ***FCS_COP.1(2/OE)(Cryptographic Signature),***
- ***FCS_COP.1(3/OE)(Cryptographic Hashing),***] cryptographic operations.

5.1.3.3 Cryptographic Key Generation (FCS_CKM.1/TSF)

FCS_CKM.1.1(TSF)

Refinement: The TSF shall [***generate***] asymmetric cryptographic keys in accordance with the specified key generation algorithm: [

- ***RSA schemes using cryptographic key sizes of 2048-bit or greater that meet the following: FIPS PUB 186-4, 'Digital Signature Standard (DSS)', Appendix B.3,***
- ***ECC schemes using 'NIST curves' [P-256, P-384, P-521] that meet the following: FIPS PUB 186-4, 'Digital Signature Standard (DSS)', Appendix B.4,***
- ***FFC schemes using cryptographic key sizes of 2048-bit or greater that meet the following: FIPS PUB 186-4, 'Digital Signature Standard (DSS)', Appendix B.1***]

and specified cryptographic key sizes [**2048 bits**].

5.1.3.4 Cryptographic Key Generation (FCS_CKM.1/OE)

FCS_CKM.1.1(OE)

Refinement: The TSF shall [***invoke interfaces provided by the Operational Environment to generate***] asymmetric cryptographic keys in accordance with the specified key generation algorithm: [

- ***RSA schemes using cryptographic key sizes of 2048-bit or greater that meet the following: FIPS PUB 186-4, 'Digital Signature Standard (DSS)', Appendix B.3,***
- ***ECC schemes using 'NIST curves' [P-256, P-384, P-521] that meet the following: FIPS PUB 186-4, 'Digital Signature Standard (DSS)', Appendix B.4,***]

and specified cryptographic key sizes [**2048 bits**].

5.1.3.5 Cryptographic Key Establishment (FCS_CKM.2/TSF)

FCS_CKM.2.1(TSF)

Refinement: The TSF shall [***perform***] key establishment in accordance with a specified cryptographic key establishment algorithm [

- ***RSA-based key establishment schemes that meet the following: NIST Special Publication 800-56B Revision 1, 'Recommendation for Pair-Wise Key Establishment Schemes Using Integer Factorization Cryptography',***
- ***Elliptic curve-based key establishment schemes that meet the following: NIST Special Publication 800-56A Revision 2, 'Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography',***
- ***Finite field-based key establishment schemes that meet the following: NIST Special Publication 800-56A Revision 2, 'Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography',***].

5.1.3.6 Cryptographic Key Destruction (FCS_CKM_EXT.4/TSF)

FCS_CKM_EXT.4.1(TSF)

The TSF shall [***destroy***] all cryptographic keys and critical security parameters which are not permanently protected from export by hardware when no longer required, in accordance with the specified cryptographic key destruction method [

- ***for volatile memory, the destruction shall be executed by a [***
  - ***single direct overwrite consisting of [zeroes]],***

].

FCS_CKM_EXT.4.2(TSF)

The TSF shall [***destroy***] all plaintext keying material cryptographic security parameters when no longer needed.

5.1.3.7 Cryptographic Operation (AES Encryption/Decryption) (FCS_COP.1(1/TSF)

FCS_COP.1.1(1/TSF)

Refinement: The TSF shall [***perform***] encryption and decryption in accordance with a specified cryptographic algorithm: [

- ***AES-CBC (as defined in NIST SP 800-38A) mode,***
- ***AES-GCM (as defined in NIST SP 800-38D) mode,***]

and cryptographic key size [***selection: 128-bit, 256-bit***].

5.1.3.8 Cryptographic Operation (AES Encryption/Decryption) (FCS_COP.1(1/OE))

FCS_COP.1.1(1/OE)

Refinement: The TSF shall [***interfaces in the operational environment to perform***] encryption and decryption in accordance with a specified cryptographic algorithm: [

- ***AES-CBC (as defined in NIST SP 800-38A) mode,***
- ***AES Key Wrap (KW) (as defined in NIST SP 800-38F) mode***]

and cryptographic key size [***128-bit, 256-bit***].

5.1.3.9 Cryptographic Operation (Cryptographic Signature) (FCS_COP.1(2/TSF))

FCS_COP.1.1(2/TSF)

Refinement: The TSF shall [***perform***] cryptographic signature services in accordance with the following specified cryptographic algorithms [

- ***selection: RSA Digital Signature Algorithm (rDSA) with a key size (modulus) of [2048, 3072, 4096] that meets FIPS-PUB 186-4, 'Digital Signature Standard',***
- ***Elliptic Curve Digital Signature Algorithm (ECDSA) with a key size of 256 bits or greater that meets FIPS PUB 186-4, 'Digital Signature Standard' with 'NIST curves' P-256, P-384 and [P-521] (as defined in FIPS PUB 186-4, 'Digital Signature Standard'),***].

5.1.3.10 Cryptographic Operation (Cryptographic Signature) (FCS_COP.1(2/OE))

FCS_COP.1.1(2/OE)

Refinement: The TSF shall [***invoke interfaces in the operational environment to perform***] cryptographic signature services in accordance with the following specified cryptographic algorithms [

- ***selection: RSA Digital Signature Algorithm (rDSA) with a key size (modulus) of [2048, 3072, 4096] that meets FIPS-PUB 186-4, 'Digital Signature Standard',***
- ***Elliptic Curve Digital Signature Algorithm (ECDSA) with a key size of 256 bits or greater that meets FIPS PUB 186-4, 'Digital Signature Standard' with 'NIST curves' P-256, P-384 and [P-521] (as defined in FIPS PUB 186-4, 'Digital Signature Standard'),***].

5.1.3.11 Cryptographic Operation (Cryptographic Hashing) (FCS_COP.1(3/TSF))

FCS_COP.1.1(3/TSF)

Refinement: The TSF shall [***perform***] cryptographic hashing services in accordance with a specified cryptographic algorithm [***SHA-1, SHA-256, SHA-384***] and message digest sizes [***160, 256, 384***] bits that meet the following: FIPS Pub 180-4, 'Secure Hash Standard'.

5.1.3.12 Cryptographic Operation (Cryptographic Hashing) (FCS_COP.1(3/OE))

FCS_COP.1.1(3/OE)

Refinement: The TSF shall [***invoke interfaces in the operational environment to perform***] cryptographic hashing services in accordance with a specified cryptographic algorithm [***SHA-1, SHA-256, SHA-384***] and message digest sizes [***160, 256, 384***] bits that meet the following: FIPS Pub 180-4, 'Secure Hash Standard'.

5.1.3.13 Cryptographic Operation (Keyed-Hash Message Authentication) (FCS_COP.1(4/TSF))

FCS_COP.1.1(4/TSF)

Refinement: The TSF shall [***perform***] keyed hash message authentication in accordance with a specified cryptographic algorithm HMAC-[***SHA-1, SHA-256, SHA-384***], key size [**160, 256, 384**] , and message digest sizes [***160, 256, 384***] bits that meet the following: FIPS Pub 198-1,"The Keyed Hash Message Authentication Code', FIPS Pub 180-4, 'Secure Hash Standard'.

5.1.3.14 HTTPS Protocol (FCS_HTTPS_EXT.1)

FCS_HTTPS_EXT.1.1

The TSF shall implement the HTTPS protocol that complies with RFC 2818.

FCS_HTTPS_EXT.1.2

The TSF shall implement HTTPS using TLS.

5.1.3.15 Cryptographic Random Bit Generation (FCS_RBG_EXT.1/TSF)

FCS_RBG_EXT.1.1(TSF)

The TSF shall [***perform***] all deterministic random bit generation (RBG) services in accordance with NIST Special Publication 800-90A using [***Hash_DRBG (any)***].

FCS_RBG_EXT.1.2(TSF)

The deterministic RBG shall be seeded by an entropy source that accumulates entropy from [***an Operational Environment-based noise source***] with a minimum of [***256 bits***] of entropy at least equal to the greatest security strength (according to NIST SP 800-57) of the keys and authorization factors that it will generate.

5.1.3.16 Cryptographic Random Bit Generation (FCS_RBG_EXT.1/OE)

FCS_RBG_EXT.1.1(OE)

The TSF shall [***invoke interfaces in the operational environment to perform***] all deterministic random bit generation (RBG) services in accordance with NIST Special Publication 800-90A using [***CTR_DRBG (AES)***].

FCS_RBG_EXT.1.2(OE)

The deterministic RBG shall be seeded by an entropy source that accumulates entropy from [***an Operational Environment-based noise source***] with a minimum of [***256 bits***] of entropy at least equal to the greatest security strength (according to NIST SP 800-57) of the keys and authorization factors that it will generate.

5.1.3.17 Cryptographic Key Storage (FCS_STG_EXT.1)

FCS_STG_EXT.1.1

Persistent private and secret keys shall be stored within the [***Operational Environment***] [***in a hardware cryptographic module***].

5.1.3.18 TLS Client Protocol with Mutual Authentication (FCS_TLSC_EXT.2)

FCS_TLSC_EXT.2.1

The TSF shall implement [***TLS 1.2 (RFC 5246), TLS 1.1 (RFC 4346)***] supporting the following ciphersuites: [

- ***TLS_DHE_RSA_WITH_AES_128_CBC_SHA as defined in RFC 3268,***
- ***TLS_DHE_RSA_WITH_AES_256_CBC_SHA as defined in RFC 3268,***
- ***TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA as defined in RFC 4492,***
- ***TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA as defined in RFC 4492,***
- ***TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5246,***
- ***TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 as defined in RFC 5246,***
- ***TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5288,***
- ***TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5288,***
- ***TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289,***
- ***TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289,***
- ***TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5289,***
- ***TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289,***
- ***TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA as defined in RFC 4492,***
- ***TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA as defined in RFC 4492,***
- ***TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5289,***
- ***TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289,***
- ***TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289,***
- ***TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289***

]. (TD0294 applied)

FCS_TLSC_EXT.2.2

The TSF shall verify that the presented identifier matches the reference identifier according to RFC 6125. (TD0294 applied)

FCS_TLSC_EXT.2.3

The TSF shall establish a trusted channel only if the peer certificate is valid. (TD0294 applied)

FCS_TLSC_EXT.2.4

The TSF shall [***present the Supported Elliptic Curves Extension in the Client Hello with the following NIST curves: [secp256r1, secp384r1, secp521r1]***] in the Client Hello. (TD0294 applied)

FCS_TLSC_EXT.2.5

The TSF shall support mutual authentication using X.509v3 certificates. (TD0294 applied)

5.1.3.19 TLS Server Protocol (FCS_TLSS_EXT.2)

FCS_TLSS_EXT.2.1

The TSF shall implement [***TLS 1.2 (RFC 5246), TLS 1.1 (RFC 4346)***] supporting the following ciphersuites: [

- ***TLS_DHE_RSA_WITH_AES_128_CBC_SHA as defined in RFC 3268,***
- ***TLS_DHE_RSA_WITH_AES_256_CBC_SHA as defined in RFC 3268,***
- ***TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA as defined in RFC 4492,***
- ***TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA as defined in RFC 4492,***
- ***TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5246,***
- ***TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 as defined in RFC 5246,***
- ***TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5288,***
- ***TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5288,***
- ***TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289,***
- ***TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289,***
- ***TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5289,***
- ***TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289,***
- ***TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA as defined in RFC 4492,***

- ***TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA as defined in RFC 4492,***
- ***TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5289,***
- ***TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289,***
- ***TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289,***
- ***TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289***

].

and no other ciphersuite. (TD0294 applied)

FCS_TLSS_EXT.2.2

The TSF shall deny connections from clients requesting SSL 2.0, SSL 3.0, TLS 1.0 and [***no other TLS versions***]. (TD0294 applied)

FCS_TLSS_EXT.2.3

The TSF shall [***generate EC Diffie-Hellman parameters over NIST curves [secp256r1, secp384r1,secp521r1] and no other curves, generate Diffie-Hellman parameters of size 2048 bits and [no other sizess]***]. (TD0294 applied)

FCS_TLSS_EXT.2.4

The TSF shall support mutual authentication of TLS clients using X.509 certificates. (TD0294 applied)

FCS_TLSS_EXT.2.5

For communications configured to require TLS with mutual authentication, the TOE shall not establish a trusted channel if the client certificate is invalid. (TD0294 applied)

FCS_TLSS_EXT.2.6

For The TSF shall respond with a fatal TLS error if the distinguished name (DN) or Subject Alternative Name (SAN) contained in a certificate presented for client authentication does not match the expected identifier for the client. (TD0294 applied)

.

5.1.4 User data protection (FDP)

5.1.4.1 Certificate Profiles (FDP_CER_EXT.1)

FDP_CER_EXT.1.1

The TSF shall implement a certificate profile function and shall ensure that issued certificates are consistent with configured profiles.

FDP_CER_EXT.1.2

The TSF shall generate certificates using profiles that comply with requirements for certificates as specified in IEFT RFC 5280, 'Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile'. At a minimum, the TSF shall ensure that:

a) The version field shall contain the integer 2.
b) The issuerUniqueID or subjectUniqueID fields are not populated.
c) The serialNumber shall be unique with respect to the issuing Certification Authority.
d) The validity field shall specify a notBefore value that does not precede the current time and a notAfter value that does not precede the value specified in notBefore.
e) The issuer field is not empty.
f) The signature field and the algorithm in the subjectPublicKeyInfo field shall contain the OID for a signature algorithm specified in FCS_COP.1(2).
g) The following extensions are supported:
   a. subjectKeyIdentifier
   b. authorityKeyIdentifier
   c. basicConstraints
   d. keyUsage
   e. extendedKeyUsage
   f. certificatePolicy

h) A subject field containing a null Name (e.g., a sequence of zero relative distinguished names) is accompanied by a populated critical subjectAltName extension.
i) The subjectKeyIdentifier extension is populated with a value unique for each public key contained in a certificate issued by the TSF.
j) The authorityKeyIdentifier extension in any certificate issued by the TOE must be populated and must be the same as the subjectKeyIdentifier extension contained in the TOE's signing certificate.
k) Populated keyUsage and extendedKeyUsage fields in the same certificate contain consistent values.

FDP_CER_EXT.1.3

The TSF shall be able to generate at least 20 bits of random for use in issued certificates to be included in [***serialNumber***] fields, where the random values are generated in accordance with FCS_RGB_EXT.1.

5.1.4.2 Certificate Request Matching (FDP_CER_EXT.2)

FDP_CER_EXT.2.1

The TSF shall establish a linkage from certificate requests to issued certificates.

5.1.4.3 Certificate Issuance Approval (FDP_CER_EXT.3)

FDP_CER_EXT.3.1

The TSF shall support the approval of certificates by [***CA Operations Staff***] issued according to a configured certificate profile.

5.1.4.4 Certificate Revocation List Validation (FDP_CRL_EXT.1)

FDP_CRL_EXT.1.1

A TSF that issues CRLs shall verify that all mandatory fields in any CRL issued contain values in accordance with ITU-T Recommendation X.509. At a minimum, the following items shall be validated:

a) If the version field is present, then it shall contain a 1.
b) If the CRL contains any critical extensions, then the version field shall be present and contain the integer 1.
c) If the issuer field contains a null Name (e.g., a sequence of zero relative distinguished names), then the CRL shall contain a critical issuerAltName extension.
d) The signature and signatureAlgorithm fields shall contain the OID for a digital signature algorithm in accordance with FCS_COP.1(2).
e) The thisUpdate field shall indicate the issue date of the CRL.
f) The time specified in the nextUpdate field (if populated) shall not precede the time specified in the thisUpdate field.

5.1.4.5 Certificate Status Information (FDP_CSI_EXT.1)

FDP_CSI_EXT.1.1

The TSF shall provide certificate status information whose format complies with [***ITU-T Recommendation X.509v2 CRL, the OCSP standard as defined by [RFC 6960]***].

FDP_CSI_EXT.1.2

The TSF shall support the approval of changes to the status of a certificate by [***CA operations staff***].

5.1.4.6 Basic Internal Transfer Protection (FDP_ITT.1)

FDP_ITT.1.1

The TSF shall prevent the disclosure, modification of user data when it is transmitted between physically separated parts of the TOE through the use of [***TLS***].

5.1.4.7 OCSP Basic Response Generation (FDP_OCSPG_EXT.1)

FDP_OCSPG_EXT.1.1

The TSF shall ensure that all mandatory fields in the OCSP response contain values in accordance with the standards specified in FDP_CSI_EXT.1. At a minimum, the following items shall be enforced:

a) The version field shall indicate current version.
b) The signatureAlgorithm field shall contain the object identifier (OID) for a digital signature algorithm in accordance with FCS_COP.1(2).
c) The thisUpdate field shall indicate the time at which the status being indicated is known to be correct.
d) The producedAt field shall indicate the time at which the OCSP responder signed the response.
e) The time specified in the nextUpdate field (if populated) shall not precede the time specified in the thisUpdate field.

5.1.4.8 Subset Residual Information Protection (FDP_RIP.1)

FDP_RIP.1.1

The TSF and [***Operational Environment***] shall ensure that any previous information content of a resource is made unavailable upon the [***deallocation of the resource from***] the following objects: [***TLS data buffer, private keys, password, secret data***].

5.1.5 Identification and authentication (FIA)

5.1.5.1 Certificate Management over CMS (CMC) Server (FIA_CMCS_EXT.1)

FIA_CMCS_EXT.1.1

The TSF shall be able to accept and process CMC full requests and [***simple requests***].

FIA_CMCS_EXT.1.2

The TSF shall be able to generate CMC simple responses and [***CMC full responses***] that are consistent with the selected certificate profile and which are in accordance with RFC 5272 as updated by RFC 6402, meeting the compliance requirements for CMS server and certification authorities in accordance with RFC 5474 as updated by RFC 6402.

FIA_CMCS_EXT.1.3

The TSF shall require CMC transport over HTTPS for online CMC messages in accordance with RFC 5273 as updated by RFC 6402, where the HTTPS is established in accordance with FCS_HTTPS_EXT.1. For CMC requests containing certificate requests other than initial certificate requests authenticated using shared secrets in AuthenticatedData requests or in the Identity Proof Version 2 Control of SignedData requests, the TSF shall require HTTPS with client authentication, shall ensure the authenticating entity is the same as the entity signing the CMC request and any subject indicated in the requested certificate(s) are the same as the authenticating entity, or the authenticating entity is [***an authorized RA for the requested subject, an AOR registered for the requested subject***].

FIA_CMCS_EXT.1.4

The TSF shall require CMC simple and full messages use cryptographic support in accordance with this profile. At a minimum the TSF shall ensure:

· Signature generation and verification for SignedData are performed in accordance with FCS_COP.1(2)
· Encryption for EnvelopedData is performed in accordance with FCS_COP.1(1)
· PasswordRecipientInfo for EnvelopedData or AuthenticatedData is derived in accordance with FCS_COP.1(5)
· hashAlgId in Identity Proof Version 2 control, keyGenAlgorithm in Pop Link Witness Version 2 control, witnessAlgID in Encrypted POP and Decrypted POP

controls, hashAlgorithm in Publish Trust Anchors control are in accordance with FCS_COP.1(3)

· macAlgId in Identity Proof Version 2 control, macAlgorithm in POP Link Witness Version 2 Control, and the POPAlgID in Encrypted POP and Decrypted POP controls, are in accordance with FCS_COP.1(4)

· DHPOP mechanisms shall be as specified in RFC 6955 with cryptographic support in accordance with this Protection Profile

FIA_CMCS_EXT.1.5

The TSF shall accept, process and export CMC messages under the control of local privileged user sessions for privileged users with CA Operations Staff, [***no other]*** role.

5.1.5.2 Certificate Enrollment (FIA_ENR_EXT.1)

FIA_ENR_EXT.1.1

The TSF shall be able to generate a certificate request to an external certification authority to receive a CA certificate for a CA's signing key using [***PKCS#10 in accordance with FIA_X509_EXT.3***].

5.1.5.3 Authentication Mechanism (FIA_UAU_EXT.1)

FIA_UAU_EXT.1.1

The TSF shall [***provide***] a [***[certificate-based authentication mechanism]***] to perform privileged user authentication.

5.1.5.4 User Identification and Authentication (FIA_UIA_EXT.1)

FIA_UIA_EXT.1.1

The TSF shall allow the following actions prior to requiring a non-TOE entity to initiate the identification and authentication process:

- Display the warning banner in accordance with FTA_TAB.1;
- Obtain certificate status information;
-[Download certificate from repository;
- ***[Check the status of a certificate request;***
- ***List certificates from the repository;***
- ***Search for certificates in the repository;***
- ***Request a certificate using a CMC shared secret]***].

FIA_UIA_EXT.1.2

The TSF shall require each user to be successfully identified and authenticated before allowing any other TSF-mediated actions on behalf of that user, including subscriber certificate renewal, subscriber revocation requests, privileged user access, [***no other actions***].

FIA_UIA_EXT.1.3

For subscriber actions, the TSF shall verify that the DN of the certificate presented by the subscriber for authentication matches that of the certificate being affected by the subscriber's actions.

5.1.5.5 Certificate Validation (FIA_X509_EXT.1)

FIA_X509_EXT.1.1

The TSF shall [***interface with the Operational Environment to validate***] certificates in accordance with the following rules:

· IETF RFC 5280 certificate validation and certificate path validation.

· The certificate path must terminate with a certificate in the Trust Anchor Database.

· The TSF shall validate a certificate path by ensuring the presence of the basicConstraints extension and that the cA flag is set to TRUE for all CA certificates.

· The TSF shall validate the revocation status of the certificate using [***the Online Certificate Status Protocol (OCSP) as specified in FDP_CSI_EXT.1, a Certificate Revocation List (CRL) as specified in FDP_CSI_EXT.1***].

· The TSF shall validate the extendedKeyUsage field according to the following rules:

o Certificates used for trusted updates and executable code integrity verification shall have the Code Signing purpose (id-kp 3 with OID 1.3.6.1.5.5.7.3.3),

o Client certificates presented for TLS shall have the Client Authentication purpose (id-kp 1 with OID 1.3.6.1.5.5.7.3.2) in the extendedKeyUsage field,

o Server certificates presented for TLS shall have the Server Authentication purpose (id-kp 1 with OID 1.3.6.1.5.5.7.3.1) in the extendedKeyUsage field.

FIA_X509_EXT.1.2

The TSF shall only treat a certificate as a CA certificate if the basicConstraints extension is present and the CA flag is set to TRUE.

5.1.5.6 Certificate-Based Authentication (FIA_X509_EXT.2)

FIA_X509_EXT.2.1

The TSF shall [***interface with the Operational Environment to use***] X.509v3 certificates as defined by RFC 5280 to support [***TLS, HTTPS***], and [***no additional uses***]. (TD0276 applied)

FIA_X509_EXT.2.2

When the TSF cannot determine the current revocation status of a certificate, the TSF shall [not ***accept the certificate***].

FIA_X509_EXT.2.3

The TSF shall not establish a trusted communication channel if the peer certificate is deemed invalid.

5.1.5.7 X509 Certificate Request (FIA_X509_EXT.3)

FIA_X509_EXT.3.1

The TSF shall generate a Certificate Request Message as specified by RFC 2986 and be able to provide the following information in the request: public key, CA's distinguished name, [***no other information***].

FIA_X509_EXT.3.2

The TSF shall validate the chain of certificates from the Root CA upon receiving the CA Certificate Response.

5.1.6 Security management (FMT)

5.1.6.1 Management of Security Functions Behavior (Administrator Functions) (FMT_MOF.1(1))

FMT_MOF.1(1).1

Refinement: The [***TSF, Operational Environment***] shall restrict the ability to
1. manage the TOE locally and remotely;
2. configure the audit mechanism;
3. configure and manage certificate profiles;
4. modify revocation configuration;
5. perform updates to the TOE;
6. perform on-demand integrity tests;
7. import and remove X.509v3 certificates into/from the Trust Anchor Database;
[***9. configure certificate revocation list function;***
10. configure OCSP function;

13. export PKCS#10 certificate request]
to Administrators. (TD0278 applied)

5.1.6.2 Management of Security Functions Behavior (CA/RA Functions) (FMT_MOF.1(2))

FMT_MOF.1(2).1

Refinement: The [***TSF***] shall restrict the ability to
1. approve and execute the issuance of certificates;
2. configure subscriber self-service request constraints;
[***3. configure automated certificate approval management***]
to [***CA Operations Staff***].

5.1.6.3 Management of Security Functions Behavior (CA Operations Functions) (FMT_MOF.1(3))

FMT_MOF.1(3).1

Refinement: The [***TSF***] shall restrict the ability to
1. approve certificate revocation;
[***7. no other function.***]
to CA Operations Staff.

5.1.6.4 Management of Security Functions Behavior (Admin/Officer Functions) (FMT_MOF.1(4))

FMT_MOF.1(4).1

Refinement: The [***TSF, Operational Environment***] shall restrict the ability to
1. perform destruction of sensitive data when no longer needed;
[***5. no other functions***]
to [***Administrators***]. (TD0375 applied)

5.1.6.5 Management of Security Functions Behavior (Auditor Functions) (FMT_MOF.1(5))

FMT_MOF.1(5).1

The [***TSF***] shall restrict the ability to:
Delete entries from the audit trail
[***Search the audit trail***]
to auditors.

5.1.6.6 Management of TSF Data (FMT_MTD.1)

FMT_MTD.1.1

The TSF shall restrict the ability to manage the TSF data to privileged users.

5.1.6.7 Specification of Management Functions (FMT_SMF.1)

FMT_SMF.1.1

The [***TSF, Operational Environment***] shall be capable of performing the following management functions:
1. Ability to manage the TOE locally and remotely;
2. Ability to perform updates to the TOE;
3. Ability to perform archival and recovery;
4. Ability to manage the audit mechanism;
5. Ability to configure and manage certificate profiles;
6. Ability to approve and execute the issuance of certificates;
7. Ability to approve certificate revocation;
8. Ability to modify revocation configuration;
9. Ability to configure subscriber self-service request constraints;
10. Ability to perform on-demand integrity tests;
11. Ability to destroy sensitive user data when no longer needed;

12. Ability to import and remove X.509v3 certificates into/from the Trust Anchor Database;
[*13. Ability to configure the NPE ruleset;*
14. Ability to configure automated process used to approve the revocation of a certificate or information about the revocation of a certificate;
15. Ability to approve rulesets that govern the authorizations of RAs or AORs to manage particular certificates on behalf of an organization;
16. [Ability to modify the CRL configuration, Ability to modify the OCSP configuration];
17. Ability to configure the list of TOE-provided services available before an entity is identified and authenticated, as specified in FIA_UIA_EXT.1;
18. Ability to configure the cryptographic functionality;
19. Ability to import private keys;
20. Ability to export TOE private keys (not for archival);
21. Ability to disable deprecated algorithms;
22. Ability to accept certificates whose validity cannot be determined;
23. Ability to accept, process and export CMC messages;].

5.1.6.8 Restrictions on Security Roles (FMT_SMR.2)

FMT_SMR.2.1

The TSF and [***no other component***] shall maintain the roles:
- Administrator,
- Auditor,
- CA Operations Staff,
- [***no other roles***].

FMT_SMR.2.2

The TSF and [***no other component***] shall be able to associate users with roles.

FMT_SMR.2.3

The TSF and [***no other component***] shall ensure that the conditions
- No identity is authorized to assume both an Auditor role and any of the other roles in FMT_SMR.2.1; and
- No identity is authorized to assume both a CA Operations Staff role and any of the other roles in FMT_SMR.2.1

are satisfied.

Application Note: This TOE implements a role called "Agent" that is equivalent to the "CA Operations Staff" role defined in the protection profile. Therefore, references throughout this ST and other evaluation evidence to "Agent" should be interpreted as a reference to the PP defined "CA Operations Staff" role.

5.1.7 Protection of the TSF (FPT)

5.1.7.1 Failure with Preservation of Secure State (FPT_FLS.1)

FPT_FLS.1.1

The TSF shall preserve a secure state when the following types of failures occur: [***[self-test failures]***].

5.1.7.2 Basic Internal TSF Data Transfer Protection (FPT_ITT.1)

FPT_ITT.1.1

The TSF shall protect TSF data from disclosure, modification when it is transmitted between separate parts of the TOE through the use of [***TLS***].

5.1.7.3 No Plaintext Key Export (FPT_KST_EXT.1)

FPT_KST_EXT.1.1

The TSF and [***Operational Environment***] shall prevent the plaintext export of [***all keys used by the TSF***].

5.1.7.4 TSF Key Protection (FPT_KST_EXT.2)

FPT_KST_EXT.2.1

The TSF and [***Operational Environment***] shall prevent unauthorized use of all TSF private and secret keys.

5.1.7.5 Manual Trusted Recovery (FPT_RCV.1)

FPT_RCV.1.1

After [**failure of self-tests**] the TSF shall enter a maintenance mode where the ability to return to a secure state is provided.

5.1.7.6 Protection of Keys (FPT_SKP_EXT.1)

FPT_SKP_EXT.1.1

The TSF shall [***interface with the Operational Environment to implement***] the ability to prevent reading of all pre-shared keys, private, and secret keys (e.g., KEKs, DEKs, session keys).

5.1.7.7 Split Knowledge Procedures (FPT_SKY_EXT.1(2)/OTH)

FPT_SKY_EXT.1(2).1/OTH

The TSF shall [***support***] split knowledge procedures to enforce two-party control for the export of [***user private keys***] using [***RSA-based key archival***]. (TD0328 applied)

5.1.7.8 Reliable Time Stamps (FPT_STM.1)

FPT_STM.1.1

The TSF shall [***interface with the Operational Environment to provide***] reliable time stamps.

5.1.7.9 Trusted Update (FPT_TUD_EXT.1)

FPT_TUD_EXT.1.1

The TSF shall [***interface with the Operational Environment to implement***] the ability to check for updates and patches to the TOE.

FPT_TUD_EXT.1.2

The TSF shall [***interface with the Operational Environment to implement***] the ability to provide Administrators the ability to initiate updates to TOE firmware/software.

FPT_TUD_EXT.1.3

The TSF shall [***interface with the Operational Environment to implement***] the ability to verify firmware/software updates to the TOE using a digital signature prior to installing those updates.

FPT_TUD_EXT.1.4

The TSF shall [***interface with the Operational Environment to implement***] the ability to verify the digital signature whenever the software or firmware is externally loaded into the TOE and if verification fails, the TSF shall [***reject the software update***].

5.1.8 TOE access (FTA)

5.1.8.1 TSF-Initiated Termination (FTA_SSL.3)

FTA_SSL.3.1

Refinement: The TSF shall terminate a remote interactive session after a [***an Administrator-configurable time interval of session inactivity***].

5.1.8.2 User-Initiated Termination (FTA_SSL.4)

FTA_SSL.4.1

The TSF shall [***interface with the Operational Environment to implement***] the ability to allow privileged user-initiated termination of the privileged user's own interactive session.

5.1.8.3 Default TOE Access Banners (FTA_TAB.1)

FTA_TAB.1.1

Before establishing a privileged user session the TSF shall display an Administrator-configured advisory notice and consent warning message regarding use of the TOE.

5.1.9 Trusted path/channels (FTP)

5.1.9.1 Inter-TSF Trusted Channel (FTP_ITC.1)

FTP_ITC.1.1

The TSF shall use [***TLS***] to provide a trusted communication channel between itself and external network based IT entities supporting the following capabilities: [***directory services***] that is logically distinct from other communication channels and provides assured identification of its end points and protection of the channel data from modification or disclosure.

FTP_ITC.1.2

The TSF shall permit the TSF, the authorized IT entities to initiate communication via the trusted channel.

FTP_ITC.1.3

The TSF shall initiate communication via the trusted channel for [***directory services***].

5.1.9.2 Trusted Path (FTP_TRP.1)

FTP_TRP.1.1

The TSF shall use [***HTTPS, TLS***] to provide a trusted communication path between itself and remote subscribers and privileged users that is logically distinct from other communication paths and provides assured identification of its end points and protection of the communicated data from modification, disclosure.

FTP_TRP.1.2

The TSF shall permit remote subscribers and privileged users to initiate communication via the trusted path.

FTP_TRP.1.3

The TSF shall require the use of the trusted path for initial subscriber and privileged user authentication and all remote administration actions.

5.2 TOE Security Assurance Requirements

The SARs for the TOE are the components as specified in Part 3 of the Common Criteria. Note that the SARs have effectively been refined with the assurance activities explicitly defined in association with both the SFRs and SARs.

Requirement Class	Requirement Component

ADV: Development	ADV_FSP.1: Basic functional specification
AGD: Guidance documents	AGD_OPE.1: Operational user guidance
	AGD_PRE.1: Preparative procedures
ALC: Life-cycle support	ALC_CMC.1: Labelling of the TOE
	ALC_CMS.1: TOE CM coverage
ATE: Tests	ATE_IND.1: Independent testing - conformance
AVA: Vulnerability assessment	AVA_VAN.1: Vulnerability survey

Table 5-3 Assurance Components

5.2.1 Development (ADV)

5.2.1.1 Basic functional specification (ADV_FSP.1)

ADV_FSP.1.1d

The developer shall provide a functional specification.

ADV_FSP.1.2d

The developer shall provide a tracing from the functional specification to the SFRs.

ADV_FSP.1.1c

The functional specification shall describe the purpose and method of use for each SFR-enforcing and SFR-supporting TSFI.

ADV_FSP.1.2c

The functional specification shall identify all parameters associated with each SFR-enforcing and SFR-supporting TSFI.

ADV_FSP.1.3c

The functional specification shall provide rationale for the implicit categorization of interfaces as SFR-non-interfering.

ADV_FSP.1.4c

The tracing shall demonstrate that the SFRs trace to TSFIs in the functional specification.

ADV_FSP.1.1e

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

ADV_FSP.1.2e

The evaluator shall determine that the functional specification is an accurate and complete instantiation of the SFRs.

5.2.2 Guidance documents (AGD)

5.2.2.1 Operational user guidance (AGD_OPE.1)

AGD_OPE.1.1d

The developer shall provide operational user guidance.

AGD_OPE.1.1c

The operational user guidance shall describe, for each user role, the user-accessible functions and privileges that should be controlled in a secure processing environment, including appropriate warnings.

AGD_OPE.1.2c

The operational user guidance shall describe, for each user role, how to use the available interfaces provided by the TOE in a secure manner.

AGD_OPE.1.3c

The operational user guidance shall describe, for each user role, the available functions and interfaces, in particular all security parameters under the control of the user, indicating secure values as appropriate.

AGD_OPE.1.4c

The operational user guidance shall, for each user role, clearly present each type of security-relevant event relative to the user-accessible functions that need to be performed, including changing the security characteristics of entities under the control of the TSF.

AGD_OPE.1.5c

The operational user guidance 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.

AGD_OPE.1.6c

The operational user guidance shall, for each user role, describe the security measures to be followed in order to fulfil the security objectives for the operational environment as described in the ST.

AGD_OPE.1.7c

The operational user guidance shall be clear and reasonable.

AGD_OPE.1.1e

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

5.2.2.2 Preparative procedures (AGD_PRE.1)

AGD_PRE.1.1d

The developer shall provide the TOE including its preparative procedures.

AGD_PRE.1.1c

The preparative procedures shall describe all the steps necessary for secure acceptance of the delivered TOE in accordance with the developer's delivery procedures.

AGD_PRE.1.2c

The preparative procedures shall describe all the steps necessary for secure installation of the TOE and for the secure preparation of the operational environment in accordance with the security objectives for the operational environment as described in the ST.

AGD_PRE.1.1e

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

AGD_PRE.1.2e

The evaluator shall apply the preparative procedures to confirm that the TOE can be prepared securely for operation.

5.2.3 Life-cycle support (ALC)

5.2.3.1 Labelling of the TOE (ALC_CMC.1)

ALC_CMC.1.1d

The developer shall provide the TOE and a reference for the TOE.

ALC_CMC.1.1c

The TOE shall be labelled with its unique reference.

ALC_CMC.1.1e

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

5.2.3.2 TOE CM coverage (ALC_CMS.1)

ALC_CMS.2.1d

The developer shall provide a configuration list for the TOE.

ALC_CMS.2.1c

The configuration list shall include the following: the TOE itself; and the evaluation evidence required by the SARs.

ALC_CMS.2.2c

The configuration list shall uniquely identify the configuration items.

ALC_CMS.2.1e

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

5.2.4 Tests (ATE)

5.2.4.1 Independent testing - conformance (ATE_IND.1)

ATE_IND.1.1d

The developer shall provide the TOE for testing.

ATE_IND.1.1c

The TOE shall be suitable for testing.

ATE_IND.1.1e

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

ATE_IND.1.2e

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

5.2.5 Vulnerability assessment (AVA)

5.2.5.1 Vulnerability survey (AVA_VAN.1)

AVA_VAN.1.1d

The developer shall provide the TOE for testing.

AVA_VAN.1.1c

The TOE shall be suitable for testing.

AVA_VAN.1.1e

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

AVA_VAN.1.2e

The evaluator shall perform a search of public domain sources to identify potential vulnerabilities in the TOE.

AVA_VAN.1.3e

The evaluator shall conduct penetration testing, based on the identified potential vulnerabilities, to determine that the TOE is resistant to attacks performed by an attacker possessing Basic attack potential.

6. TOE Summary Specification

This chapter describes the security functions:

- Security audit
- Communication
- Cryptographic support
- User data protection
- Identification and authentication
- Security management
- Protection of the TSF
- TOE access
- Trusted path/channels

6.1 Security audit

The Security audit function satisfies the following security functional requirements:

FAU_ADP_EXT.1: The TOE generates audit records for CA related audit events itself and the TOE's Operational Environment (RHEL) triggers lower-level TLS communication events to be logged from the TOE. The TOE's Operational Environment also provides auditing for time changes, TOE updates, and attempts to delete audit data.

The TOE has different types of audit logs:

1) Signed audit logs specific to each subsystem. These signed audit logs may be found in the subsystem's audit directory (in `/var/lib/pki/<pki_instance_name>/<subsystem>/logs/signedAudit/` directory), and these logs contain audit records from that subsystem.
2) Certificate issuance data (unsigned, but protected within an internal LDAP database).

FAU_GCR_EXT.1: The TOE uses Red Hat Directory Server as its database for storing information such as certificates, CRLs, certificate requests, user definitions, each user's role, and other internal information. The TOE subsystems communicate with the LDAP database securely through TLS protected network communication utilizing X509-based client authentication.

FAU_GEN.1: The TOE minimally generates the events listed in **Table 5-2 - Auditable Events** above and includes the date, time, event type, subject, success or failure, as well as any additional content listed in **Table 5-2 - Auditable Events**.

During the installation process the TSF invokes the HSM to generate all non-ephemeral keys that will be used by the TSF. The TOE does not generate further non-ephemeral keys after installation. Because all generation occurs during installation and prior to the TSF audit mechanism becoming operational it cannot generate audits of such activity.

FAU_GEN.2: The TOE 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.1/3: The TOE provides an 'Auditors' default group and stipulates that an auditor can view the signed audit logs and is created to audit the operation of the system. The auditor cannot administer the server in any way.

FAU_STG.1(1): The TOE writes audit records into platform provide files. The TOE protects audit records using access controls that allow only an Auditor to download the audit log through TOE provided interfaces. The OE also protects audit records it stores for the TOE using file permissions. These file permissions prevent unauthorized deletion of audit records. The TOE provides no method for administrators to modify or delete certificate database data from RHCS. The TOE provides additional assurance that audit records are not modified by digitally signing audit record buffers as they are flushed into the non-volatile audit log storage that is provided by the OE. If audit log signing is enabled, then the logs can be verified by the auditors after downloading the files to the client or verified by the admin directly on the server-side without downloading the files.

FAU_STG.4: There are events that could cause the audit logging function to fail such that the TOE cannot write events to the log file in the RHEL OE. For example, audit logging could fail if the file system containing the audit log file is full or if the file permissions for the log file were accidentally changed. If audit logging fails, the each instance of a TOE subsystem that is affected will shutdown in the following manner.

- Servlets are disabled and will not process new requests.
- All pending and new requests are killed.
- The subsystem is shut down.

When this happens, administrators and auditors should work together with the operating system administrator to resolve the disk space or file permission issues. When any problem(s) is resolved, the auditor should make sure that the last audit log entries are signed. If not, they should be preserved by manual signing, archiving, and removing to prevent audit verification failures in the future. When this is completed, the administrators can restart the TOE.

FAU_STG_EXT.1: The TOE utilizes local storage in the `/var/log/pki/` directory of its RHEL operating system to store configuration data and audit log data.

6.2 Communication

The Communication function satisfies the following security functional requirements:

FCO_NRO_EXT.2: The TOE provides proof of origin for issued certificates by signing each issued certificate. The TOE further provides proof of origin for issued certificate status information by signing issued CRLs and OCSP responses. The TOE uses its HSM as a PKCS#11 token to handle the sensitive signing operations requiring the TOE's CA private keys.

The TOE verifies certificate requests, public keys contained in certificate requests, and revocation requests by requiring and verifying CMC signatures. CMC requests must bear valid proof of origin, and verifies the CMC digital signature included with the request using the public key in the request. The TOE rejects requests bearing an invalid signature.

FCO_NRR_EXT.2: The TOE provides signed CMC responses to prove receipt of request messages. Similar to the above description, the TOE includes its signature on CMC responses, thus proving that the generated CMC response originated from the TOE itself. To do this, the TOE again uses its HSM as a PKCS#11 token to use its CA keys with the HSM to sign the CMC responses.

6.3 Cryptographic support

The TOE's NSS software cryptographic module (a RHEL component of the TOE) implements cryptography used by the TOE's for TLS/HTTPS and for Trusted Updates.

The TOE's Tomcat engine uses NSS for TLS/HTTPS operations (via Tomcat JSS, and Java Security Services, and then ultimately provided by NSS). The TOE's RPM executable uses NSS to verify the signatures on RPM update packages during installation. Finally, the TOE also obtains cryptography indirectly from its Operational Environment (from a Hardware Security Module, or HSM, in the TOE's Operational Environment). .

SFR	Algorithm	NIST Standard	Cert#
FCS_CKM.1/TSF (Key Gen)	RSA IFC Key Generation	FIPS 186-4, RSA	3006/3013
	ECDSA ECC Key Generation	FIPS 186-4, ECDSA	1511/1515
	DSA FFC Key Generation	FIPS 186-4, DSA	1440/1442
FCS_CKM.2/TSF (Key Establishment)	RSA-based Key Exchange	Vendor affirm 800-56B SHS DRBG RSA	N/A
	ECC-based Key Exchange	SP 800-56A, CVL KAS ECC 800-135 TLS	2019/2026 2020/2027
	FFC-based Key Exchange	SP 800-56A, CVL KAS FFC 800-135 TLS	2019/2026 2020/2027
FCS_COP.1(1/TSF) (AES)	AES 128/256 CBC, GCM	FIPS 197, SP 800-38A/D/F	5588/5602
FCS_COP.1(2/TSF) (Sign/Verify)	RSA Sign/Verify	FIPS 186-4, RSA	3006/3013
	ECDSA Sign/Verify	FIPS 186-4, ECDSA	1511/1515
FCS_COP.1(3/TSF) (Hash)	SHA-1, SHA-256, SHA-384	FIPS 180-4	4491/4499
FCS_COP.1(4/TSF) (Keyed Hash)	HMAC-[SHA-1, SHA-256, SHA-384],	FIPS 198-1 & 180-4	3728/2735
FCS_RBG_EXT.1/TSF (Random)	Hash_DRBG (any)	SP 800-90A	2236/2246

Table 6-1 RHEL NSS CAVP Certificates

The TOE also obtains cryptography from its HSM that is part of the TOE Operational Environment. The HSM provides key generation, digital signature services, and encryption/decryption (for archival). The HSM used during evaluation testing was the FIPS 140-2 validated Thales nShield Connect 6000+ (FIPS Cert 2638) which is operated in FIPS mode and which has the CAVP certificates shown in Table 6-2.

SFR	Algorithm	NIST Standard	Thales
FCS_CKM.1/OE (Key Gen)	RSA IFC Key Generation	FIPS 186-4, RSA	RSA 1752
	ECDSA ECC Key Generation	FIPS 186-4, ECDSA	ECDSA 695
FCS_COP.1(1/OE)	AES-CBC mode	SP 800-38A	AES 3420
	AES Key Wrap (KW)	SP 800-38F	AES 3446
FCS_COP.1(2/OE) (Sign/Verify)	RSA Sign/Verify	FIPS 186-4, RSA	RSA 1752
	ECDSA Sign/Verify	FIPS 186-4, ECDSA	ECDSA 695
FCS_COP.1(3/OE) (Hash)	SHA-1/256/384 Hashing	FIPS 180-4	SHS 2826
FCS_RBG_EXT.1/OE (Random)	CTR_DRBG(AES)	SP 800-90A	DRBG 825

Table 6-2 HSM Cryptographic Algorithms

The Cryptographic support function satisfies the following security functional requirements:

FCS_CDP_EXT.1/TSF, FCS_CDP_EXT.1/OE: The TOE includes the NSS software cryptographic module and relies upon its HSM that is part of the TOE's Operational Environment. The TOE’s NSS software module has been CAVP tested, and the TOE uses a FIPS 140-2 validated hardware cryptographic module to provide protected private key storage and functionality for CA functions. The TOE invokes the interfaces of its NSS software module through Network Security Services for Java (JSS) and the JCE provider it offers. JSS and JCE are Java JNI bridges to NSS C shared libraries. The TOE similarly accesses the HSM in its operational environment through the JSS/JCE to access the HSM as a PKCS#11 token.

FCS_CKM.1/TSF, FCS_CKM.1/OE, FCS_CKM.2/TSF: The TOE provides generation and establishment of asymmetric keys as shown below in Table 6-3.

Algorithm	Key/Curve Sizes	Usage
RSA, FIPS 186-4 Appendix B.3	2048/3072/4096	CA/PKI and TLS Authentication certificates
ECDSA, FIPS 186-4 Appendix B.4	P-256/384/521	CA/PKI and TLS Authentication certificates
DH keys and domain params, FIPS 186-4 Appendix B.1	2048/3072	TLS KeyEx (TLS/HTTPS)
ECDHE keys (not domain params), FIPS 186-4 Appendix B.4	P-256/384/521	TLS KeyEx (TLS/HTTPS)

Table 6-3 Asymmetric Key Generation

The TOE relies upon an external HSM within its Operational Environment to securely generate and store RSA and ECDSA keypairs related to certificate related functions (CA functions) as well as TLS authentication keypairs during TOE installation.

The TOE uses its NSS cryptographic library during TLS key exchange to generate ephemeral asymmetric keys during negotiation of TLS_DHE_* and TLS_ECDHE_* cipher suites. The TOS's NSS library has received NIST CAVP certificates (see the table Table 6-1 RHEL NSS CAVP Certificates above for all of the TOE'S algorithm certificates).

FCS_CKM_EXT.4/TSF: The TOE's NSS cryptographic library clears sensitive TLS session keys from memory when no longer needed (i.e., when the TLS session has ended). Clearing is done by directly overwriting the session key with zeros. Beyond transient keying material associated with TLS/HTTPS sessions, the TOE stores all other keying material securely in its external HSM. The HSM provides secure hardware storage and does not permit export of the keying material and performs cryptographic operations on behalf of TOE within its secure hardware.

FCS_COP.1(1/TSF), FCS_COP.1(1/OE) – AES Encryption/Decryption: The TOE utilizes AES encryption to protect TLS communications (in which case the TOE's NSS library provides the AES implementations) as well as to archive user keys (in which case, the TOE's external HSM provides the secure, hardware AES implementation). The HSM utilizes either AES in CBC mode for encryption and decryption of archived keys, or its implementation of AES Key Wrap.

FCS_COP.1(2/TSF), FCS_COP.1(2/OE) – Cryptographic Signature: The TOE checks signatures while performing certificate and certificate path validation during authentication of a TLS peer and during processing of CMC requests. The TOE uses its NSS library to perform these public key-based verification operations. The NSS library is capable of providing cryptographic signature services for RSA (with 2048-bit, 3072-bit and 4096-bit keys) as well as ECDSA (with NIST curves P-256, P-384 and P-521) as described by Table 6-1 RHEL NSS CAVP Certificates.

The TOE also generates signatures (using the HSM) regularly as part of its PKI operations (including CA issuance of certificates and CRLs, CMC response messages, and signing of OCSP responses). For these operations, the TOE utilizes its external HSM (which can use stored private keys to generate signatures securely within its hardware). The TOE only requests signatures using approved algorithms (RSA and ECDSA) and keysizes (2048-bit RSA or greater and ECDSA using curves P-256, P-384, and P-521) from the HSM in the TOE's OE.

Finally, the TOE verifies signatures on RPM update packages using 4096-bit RSA using its NSS library.

FCS_COP.1(3/TSF), FCS_COP.1(3/OE) – Cryptographic Hashing: The TOE uses hashing both during signature generation and verification as well as during integrity checking. The TOE performs hashing for public key operations (signature verification).

The TOE uses its external HSM to perform hashing for signature generation and uses NSS to verify RPMs before installation. The TOE only requests signatures using approved hash algorithms (SHA-1/256/384) from the HSM in the TOE's OE. The TOE uses NSS to verify RPM signatures using SHA-256. The TOE uses NSS to perform hashing as part of TLS integrity checking using SHA-1, SHA-256, SHA-384.

FCS_COP.1(4/TSF) – Keyed-Hash Message Authentication: The TOE calculates MAC integrity checksums (HMAC-SHA-1 and HMAC-SHA-256) during TLS/HTTPS operations using only its NSS library.

FCS_HTTPS_EXT.1: The TOE provides all users (including those in administrative roles of admin, agent or auditor) with an HTTPS interface to configure the TOE. The user authenticates over the HTTPS interface using an x509 certificate. The user's browser sends the user's X.509 certificate to the TOE during the TLS negotiation. If the TOE finds the user's certificate trustworthy (i.e., valid and chaining to a trusted root), then the TOE establishes the TLS session and TOE permits further attempted user actions depending upon that user's authorization. If the user is an administrative user (e.g., an admin, agent or auditor), then the TOE allows management operations available to that user. If the user is not an administrative user, then the TOE permits only end-entity functionality (i.e., public features) for that user. The TOE's NSS module provides the TLS functionality supporting HTTPS.

FCS_RBG_EXT.1(TSF): The TOE's NSS library includes a SHA-256 Hash_DRBG that the library uses for all random values needed for higher level cryptographic operations (signature generation key exchange, and TLS random values). The NSS library seeds itself from its Operational Environment-based noise source (RHEL).

FCS_STG_EXT.1: The TOE includes the following keys:

- CA private key
- Subsystem private key (each subsystem)
- TLS server (authentication) private key (each subsystem)
- Audit signing key (each subsystem)

FCS_TLSC_EXT.2: The TOE acts as a TLS client (using the NSS module) during internal communications between the different subsystems within the TOE environment and when acting as a directory services client. The TOE supports the ciphersuites listed in **Table 6-4**. The TOE supports reference identifiers of Common name, UID, DNS name, and IP addresses, but does not support wildcards in the reference identifiers. Furthermore, the TOE does not support or utilize certificate pinning. The TOE's subsystems present their subsystem certificates as a TLS client certificate during the TLS handshake authentication, and the TOE's Client Hello messages include the secp256r1, secp384r1 and secp521r1 curves in the Supported Groups Extension, and the TOE requires no administrator configuration for that extension.

Ciphersuites
TLS_DHE_RSA_WITH_AES_128_CBC_SHA as defined in RFC 3268,
TLS_DHE_RSA_WITH_AES_256_CBC_SHA as defined in RFC 3268,
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA as defined in RFC 4492,
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA as defined in RFC 4492,
TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5246,
TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 as defined in RFC 5246,
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5288,
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5288,
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289,
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289,
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5289,
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289,

TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA as defined in RFC 4492,
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA as defined in RFC 4492,
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5289,
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289,
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289,
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289

Table 6-4 Supported TLS Server & TLS Client Ciphersuites

FCS_TLSS_EXT.2: The TOE acts as a TLS server when servicing incoming requests from administrators (through browsers, through CLI tools, and through the TOE's Java console), when servicing requests from subscriber requests, and during internal communications within the TOE environment. The TOE supports the ciphersuites listed in **Table 6-4**, and the TOE denies connections attempting to use TLS version 1.0 and lower. The TOE supports generation and use of DHE with 2048-bitprimes and supports generation and use of ECDHE keys along secp256r1, secp384r1 and secp521r1 curves. While the TOE does not require any administrator configuration of the supported ECDHE curves (the TOE automatically supports negotiation of P-256/384/521), the administrator can (and must) configure the list of ciphersuites that the TOE will negotiate (e.g., ensuring that the configured ciphersuites match the TOE certificate type [RSA or ECDSA] that the administrator has configured). The TOE compares the expected reference identifier (either a server’s DNS name or a user’s UID) to the value found the presented certificate. The TOE requires an exact match of the expected reference identifier with that presented by the certificate.

The TOE authenticates TLS clients that are connecting to its TLS server interface through the use of x509 certificates. The TLS client presents its peer certificate to the TOE during the TLS negotiation. If the certificate is trustworthy (i.e., valid and chaining to a trusted root), then the TLS session is established and further attempted actions by the user may occur depending upon that user's authorization. If the peer is an administrative user (e.g., an admin, agent or auditor) or TOE subsystem (e.g., CA, OCSP responder, KRA), then requested operations available to that peer are allowed. If the peer is not an administrative user and not a TOE subsystem, then only end-entity functionality (i.e., public features) are available to that user.

6.4 User data protection

The User data protection function satisfies the following security functional requirements:

FDP_CER_EXT.1: The TOE provides administrators a customizable framework to apply policies for incoming certificate requests and to control the input request types and output certificate types; these are called certificate profiles. Certificate profiles set the required information for certificate enrollment. Every certificate request received by the TOE must specify an applicable certificate profile to be used to issue that certificate.

The TOE ensures that a certificate-requesting subject possesses the applicable private key by requiring the CMC certificate request be signed using the applicable private key. Additionally, the TOE ensures that when issuing certificates, the serialNumber field contains at least 20 random bits drawn from the TOE's NSS module.

FDP_CER_EXT.2: The TOE provides linkage between submitted certificate requests and the resulting issued certificate through the TOE's audit logs. The TOE assigns each certificate request a unique request ID and associates that request ID with the issued certificate (if issued). An administrator can search through the TOE's audit logs to identify the request ID associated with a given certificate request, and can then search using that request ID to find any certificate issued.

FDP_CER_EXT.3: The TOE provides a flexible set of rules to govern the approval of certificate requests (and the subsequent issuance of certificates). The TOE allows Agents (CA Operations Staff) to approve certificate requests, as well as allowing configuration of automatic-approval subject to certain rules. Furthermore, the TOE allows a configured TPS subsystem to submit certificate requests for issuance. The TOE's rules (profiles) can be changed by an Agent in conjunction with an Admin. First the Agent needs to disable the profile, the Admin can then make changes to the profile, and then the Agent can review, approve, and enable the modified profile.

FDP_CRL_EXT.1: The TOE supports generation and issuance of CRLs and delta CRLs. The TOE supports v2 CRLs and supports all of the PP-reference values. The TOE always includes a non-null Name in the issuer field of the CRLs it issues.

FDP_CSI_EXT.1: The TOE provides certificate status information through published CLRs and by responding to OCSP requests. Published CRLs comply with ITU-T Recommendation X.509v2 CRL, while OCSP support complies with RFC 6960 using the HTTP GET format. The TOE allows Agents to revoke a certificate through a CMC request and through the HTTPS interface.

FDP_ITT.1: The TOE utilizes TLS to secure communications between the different subsystems in the TOE's operational environment.

FDP_OCSPG_EXT.1: The TOE supports OCSP as a method for providing certificate revocation status. The TOE uses its NSS module for OCSP response processing. The TOE responds to an OCSP query with a response that includes fields defined by RFC 6960 and which include at least: version, signature Algorithm, this Update, produced At, and cert Status.

FDP_RIP.1: The TOE makes previous information content unavailable during TLS protected communications by clearing all sensitive information buffers after use and prior to releasing the data buffer memory.

6.5 Identification and authentication

The Identification and authentication function satisfy the following security functional requirements:

FIA_CMCS_EXT.1: The TOE supports CMC requests and responses. RedHat provides client-side CMC tools (and TOE support to offer the interfaces used by the client-tools) for use by administrators. These are CLI tools that create CMC messages, submit them to them to the TOE, retrieve the CMC response, and interpret the CMC response. Transmission of a CMC request between the TOE and client-peer is protected using TLS, and takes the form of an HTTPS Rest-API interface invocation. When the user already has a valid certificate (e.g., the user is an Agent requesting a certificate on behalf of another user, the user is renewing a certificate), the TOE uses that certificate to authenticate the TLS connection. Where no certificates are initially available, the TOE relies upon HTTPS with CMC Shared Secret authentication. The TOE supports enrolling subordinate CAs and the other TOE subsystems using these CMC tools.

FIA_ENR_EXT.1: The TOE, during installation, generates a PKCS#10 certificate request for enrollment with a superior CA; however, if needed, the TOE also provides administrators a command line tool to manually convert the PKCS#10 request into a CMC request.

FIA_UAU_EXT.1: The TOE's NSS module provides TLS and certificate checking; the TSF relies upon the authentication performed by NSS in order to perform authorization (role checks) to determine functions available to the user.

FIA_UIA_EXT.1: The TOE allows users to authenticate using a certificate as part of mutually authenticated TLS. Administrative users (those defined to the TOE and assigned a role) may access the TOE via any of its interfaces which include an HTTPS WebUI, a set of TLS protected command line tools, and a pkiconsole application. The command line tools and pkiconsole application both utilize HTTPS protected REST API interfaces which ultimately are protected by TLS. All interfaces require user to authenticate using a certificate. In order for the certificate to identify the administrative user, the certificate must include a UID that identifies the TOE user, and the match the certificate configured for that user.

Users with valid certificates but which are not assigned a role as well as users that do not present certificates, are considered an 'end-entity'. An End-Entity may access public functions through the HTTPS WebUI and HTTPS protected REST API interfaces.

Prior to login, all users, end-entities or IT entities contacting the TOE may

- display a warning banner or
- obtain certificate status via an OCSP lookup or downloading a CRL.

Additionally, an end-entity accessing the TOE without first authenticating within a TLS handshake, is allowed to perform the following operations.

- Download a certificate or CA cert chain from the repository.
- Check the status of a certificate request.
- List certificates from the repository.
- Search for certificates in the repository.
- Request a certificate using a CMC shared secret.
-

FIA_X509_EXT.1: The TOE relies upon its NSS library to validate certificates. The NSS Library will validate the following: the certificate is not revoked, the certificate has a valid certificate path from the peer's certificate through any intermediary CAs to a root CA trusted by the TOE, the current time is within the certificate's validity period, and the certificate includes extensions that are required or consistent with the current use (e.g., CA flag value, KU and EKU). Certificate revocation status is performed by getting the revocation status of all certificates (starting with the root certificate and working down the chain). The TOE will reject any certificate for which it cannot determine the revocation status and hence not accept the connection attempt if a revocation status cannot be obtained..

FIA_X509_EXT.2: The TOE uses certificates and trusted root CAs in different ways.

First, the TOE NSS library uses certificates for authentication in TLS. Second, the TOE uses certificates and trusted root CAs as part of its PKI (Public Key Infrastructure) operations (validating certificate requests, OCSP requests, etc.).

The TOE handles unreachable revocation servers by rejecting the certificate.

FIA_X509_EXT.3: The TOE can generate Certificate Request Messages conforming to RFC 2986. The TOE also provides a tool to validate certificates during import.

6.6 Security management

The Security management function satisfies the following security functional requirements:

FMT_MOF.1(1)/(2)/(3): Each subsystem of the TOE has an Administrator role that can perform management activities on that particular subsystem. Likewise, each subsystem has an auditor role that can perform the audit management functions on the subsystem and an Agent role (note the Agent can only perform certificate issuing tasks so the role is limited on anything but a CA). So for example, a user can be a CA Admin but not a KRA Admin. Agents can approve the issuance of certificates, configure subscriber self-service request constraints, configure automated certificate approval, and revoke issued certificates and end entities can revoke certificates issued to them. Administrators can perform the following tasks: manage the TOE locally and remotely, manage certificate profiles, perform updates, perform integrity tests, import and remove certificates from the Trust Anchor database, configure revocation parameters (CRL and OCSP), export a certificate request, and configure audit.

FMT_MOF.1(4): The TOE (through the HSM in its Operational Environment), allows backup/archival of its CA keys. The TOE also allows for the archival of user/subscriber keys through the Key Recovery Agent (KRA).

FMT_MOF.1(5): The TSF does not provide any means for administrators, agents, or auditors to delete logs, but auditors may download a copy of the audit logs in order to perform a search.

FMT_MTD.1: The TOE provides no access to TSF data to unauthenticated users.

FMT_SMF.1: The TOE provides some functions directly (i.e., the TSF provides them) and provides others through the OE [denoted below as "(OE - <component>)"].

1. Ability to manage the TOE locally and remotely;
2. Ability to perform updates to the TOE (OE - RHEL);
3. Ability to perform archival and recovery (OE - HSM);
4. Ability to manage the audit mechanism;
5. Ability to configure and manage certificate profiles;
6. Ability to approve and execute the issuance of certificates;
7. Ability to approve certificate revocation;
8. Ability to modify revocation configuration;
9. Ability to configure subscriber self-service request constraints;
10. Ability to perform on-demand integrity tests;
11. Ability to destroy sensitive user data when no longer needed (OE - LDAP);
12. Ability to import and remove X.509v3 certificates into/from the Trust Anchor Database (OE – RHEL nssdb);

[***13. Ability to configure the NPE ruleset;***
14. Ability to configure automated process used to approve the revocation of a certificate or information about the revocation of a certificate;
15. Ability to approve rulesets that govern the authorizations of RAs or AORs to manage particular certificates on behalf of an organization;
16. ["Ability to modify the CRL configuration", Ability to modify the OCSP configuration] (OE – RHEL);
17. Ability to configure the list of TOE-provided services available before an entity is identified and authenticated, as specified in FIA_UIA_EXT.1 (OE – web.xml);
18. Ability to configure the cryptographic functionality (OE – server.xml);
19. Ability to import private keys (OE – nssdb and HSM);
20. Ability to export TOE private keys (not for archival) (OE – nssdb);
21. Ability to disable deprecated algorithms (OE – server.xml);
22. Ability to accept certificates whose validity cannot be determined;
23. Ability to accept, process and export CMC messages;].

FMT_SMR.2: The TOE provides a group mechanism to create the three roles described below. A user's privileges are determined by the group membership of the user. These groups are defined and enforced by the TOE. During installation the TOE is configured to prevent a user from being a member of multiple groups by setting the multiroles.enable parameter in CS.cfg to false as required by the guidance,. Once set and restarted, users may not belong to more than one role. The following groups are created by default and used as administrative roles:

- Administrators. This group is given full access to all of the tasks available in the administrative interface.
- Agents. This group is the "CA Operations Staff" and is given full access to all of the tasks available in the agent services interface.
- Auditors. This group is given access to view the signed audit logs. This group does not have any other privileges.

The TOE also uses the group mechanism to define "Enterprise administrators" group. Each subsystem instance is automatically assigned a subsystem-specific group as an enterprise administrator when it is joined to a security domain during configuration. These groups automatically provide trusted relationships among subsystems in the security domain, so that each subsystem can efficiently carry out interactions with other subsystems. As indicated in guidance documentation, users must never be assigned to these "Enterprise administrator" groups.

The permission mechanisms provided by the OE (RHEL) must be configured to be consistent with these group definitions. Specifically, the OE must be configured to ensure that auditors have access only to audit data files, and that administrators have access to all other TOE data.

Every TOE interfaces accounts for a user's role when determining whether an action can be performed. Thus, some portion of each of the various TOE interfaces is available to every role. While some administrator operations require the use of OE functions (e.g., setting time and TOE product updates), all management operations available to agents are provided through the TOE interfaces and not by the OE. Auditors utilize TOE interfaces to view audit data, and are not required to access audit data through OE interfaces.

6.7 Protection of the TSF

The Protection of the TSF function satisfies the following security functional requirements:

FPT_FLS.1: RHCS self-tests refers to checking for the accessibility and validity of system certificates for each subsystem when the subsystem initializes. If a system certificate is invalid or unavailable the subsystem preserves a secure state by refusing to start.

All subsystem types will first check and verify (SystemCertsVerification) the validity and usage setting of all certificates that are defined (in CS.cfg) system certificates. This includes obtaining OCSP status for the certificates if so configured.

CA:

CAPresence - At startup time, when ready, check to see the CA can retrieve its CA signing cert.

CAValidity - check for validity of the CA signing cert (this seems to be already covered by SystemCertsVerification)

KRA:

KRAPresence - At startup time, when ready, check to see the the KRA can retrieve its transport cert.

OCSP:

OCSPPresence - similiar to CAPresence

OCSPValidity - similiar to CAValidity

FPT_ITT.1: The TOE utilizes TLS to protect communications between subsystems within the TOE as well as to secure administrative access. Each TOE subsystem obtains a certificate used to authenticate itself to other TOE subsystems. A subsystem certificate is presented any time a TOE subsystem is acting as a TLS client when communicating with another TOE subsystem.

FPT_KST_EXT.1: The TOE does not allow the plaintext export of any private keys. The TOE stores its private keys within an HSM that prevents export.

FPT_KST_EXT.2: The TOE relies upon its NSS library and the HSM in the Operational Environment to prevent unauthorized access to TSF's private keys and secret keys, respectively. The TOE provides no interfaces that allow users to access TSF private keys. Processes within the environment are prevented from accessing NSS databases through the use of process specific NSS databases that are restricted using process and file permissions provided by the OE. The NSS library also requires every access to the NSS database to be authenticated. The HSM does not offer any interfaces to users, and its interfaces to processes are authenticated by the HSM, and are restricted by process and file permissions provided by the OE to HSM driver operations.

FPT_RCV.1: Once RHCS has failed to start (in the event of a self-test failure) the administrator must manually correct the problem and start the server again.

FPT_SKP_EXT.1: The TOE relies upon its HSM within the Operational Environment to prevent any direct access (read or view the value of) to private keys. The HSM is used to store the private key for the following certificates:

- The CA Signing Certificate for every CA
- All TOE subsystem certificates
- The OCSP signing certificate for every OCSP subsystem
- The audit signing certificates from each subsystem

- All KRA storage and tansport certificates

Only the TOE can access these private keys, and the HSM only allows the TOE to request signing operations with private keys. The HSM will execute the signing operation and return the resulting signature to the TOE, thus preventing direct access by the TOE.

FPT_SKY_EXT.1(2)/OTH: The TOE's Key Recovery Agent (KRA) subsystem provides a mechanism to provide archival of user encryption key pairs. The TOE uses an m-of-n ACL-based recovery scheme. The TOE uses its existing access control scheme to ensure recovery agents are appropriately authenticated over TLS and requires that the agent belong to a specific recovery agent group, by default the Key Recovery Authority (KRA) Agents Group. The recovery request is executed only when m-of-n (a required number of) recovery agents have granted authorization to the request.

FPT_STM.1: The TOE uses the time during each operation involving certificates (issuing certificates, providing CMC responses, issuing CRLs, providing OCSP responses, verifying certificate chains [the notBefore and noAfter check], revoking certificates, and when renaming audit logs during log file rolling) and during audit event creation. To do this, the TOE obtains the time from its RHEL operating system.

FPT_TUD_EXT.1: The TOE relies upon its RHEL operating system to verify the signature on RHCS packages, allowing installation only if the digital signature is valid and thereby provide trusted update functionality. The RHEL yum and RPM mechanisms are utilized by RHCS to provide signed packages to provide TOE patches and updates. Signature verification of packages occurs automatically once the administrator invokes the 'yum update' command. The 'yum' mechanism then utilizes the RPM mechanism to verify the digital signatures in each RPM package. RedHat uses an RSA 4096-bit public key and a SHA-256 hash to produce the digital signature in an RPM package, and the TOE verifies the digital signature against its stored copy of the public key that is stored in a private RPM directory and protected by RHEL and SELinux mechanisms.

6.8 TOE access

The TOE access function satisfies the following security functional requirements:

FTA_SSL.3: The TOE uses a configured TLS session time out that it associates with the administrative interface into each subsystem.

FTA_SSL.4: An administrator using a web browser or the PKI console to administer the TOE can close their browser or exit the console in order to close their TLS session. When using command line tools (other than the pki console), each command represents a single, authenticated connection that the TOE terminates upon conclusion of the command.

FTA_TAB.1: The TOE provides the administrator the ability to configure a banner that the TOE will display to the administrator for before stating each interactive session on the HTTPS (WebUI), a set of TLS protected command line tools, and a pkiconsole application utilizing HTTPS protected REST API interfaces.

6.9 Trusted path/channels

The Trusted path/channels function satisfies the following security functional requirements:

FTP_ITC.1: The TOE utilizes TLS to secure communications between the TSF and an LDAP directory server in the operational environment (using LDAPS).

FTP_TRP.1: The TOE allows remote access protected by TLS. Interfaces available to administrators include an HTTPS (WebUI), a set of TLS protected command line tools, and a pkiconsole application. The command line tools and pkiconsole application both utilize HTTPS protected REST API interfaces which ultimately are protected by TLS.

The TOE allows remote access protected by TLS. Interfaces available to administrators include an HTTPS WebUI, a set of TLS protected command line tools, and a pkiconsole application. The command line tools and pkiconsole application both utilize HTTPS protected REST API interfaces which ultimately are protected by TLS.