



Pulse Secure Virtual Appliance 8.2/5.3 Security Target

Version: 3.2

April 2018

Prepared For:

Pulse Secure, LLC
2700 Zanker Road
Suite 200
San Jose, CA 95134

Prepared By:

Acumen Security

Notices:

©2018 Pulse Secure, LLC All rights reserved. All other brand names are trademarks, registered trademarks, or service marks of their respective companies or organizations

It is prohibited to copy, reproduce or retransmit the information contained within this documentation without the express written permission of 2700 Zanker Road, Suite 200, San Jose, CA 95134

Table of Contents

1.	Security Target (ST) Introduction.....	6
1.1	Security Target Reference.....	6
1.2	Target of Evaluation Reference	6
1.3	Target of Evaluation Overview.....	7
1.3.1	TOE Product Type.....	7
1.3.2	TOE Usage	7
1.3.3	TOE Major Security Features Summary	7
1.3.4	TOE IT environment hardware/software/firmware requirements.....	7
1.4	Target of Evaluation Description	9
1.4.1	Target of Evaluation Physical Boundaries.....	9
1.4.2	Target of Evaluation Logical Boundaries.....	9
1.5	Notation, formatting, and conventions	11
2.	Conformance Claims	12
2.1	Common Criteria Conformance Claims.....	12
2.2	Conformance to Protection Profiles	12
2.3	Conformance to Security Packages.....	14
2.4	Conformance Claims Rationale.....	14
3.	Security Problem Definition	16
3.1	Threats	16
3.2	Organizational Security Policies.....	17
3.3	Assumptions.....	17
4.	Security Objectives.....	19
4.1	Security Objectives for the Operational Environment.....	19
5.	Extended Components Definition.....	20
5.1	Extended Security Functional Requirements Definitions	20
5.2	Extended Security Assurance Requirement Definitions	20
6.	Security Requirements.....	21
6.1	Security Function Requirements.....	21
6.1.1	Class FAU: Security Audit	22
6.1.2	Class FCS: Cryptographic Support	26
6.1.3	Class FIA: Identification and Authentication.....	34
6.1.4	Class FMT: Security Management.....	37
6.1.5	Class FPT: Protection of the TSF.....	40
6.1.6	Class FTA: TOE Access	42

- 6.1.7 Class FTP: Trusted Path/Channels..... 43
- 6.2 Security Assurance Requirements 45
 - 6.2.1 Extended Security Assurance Requirements 45
 - 6.2.1.1 ASE: Security Target 45
- 7. TOE Summary Specification 52
 - 7.1 Security Audit..... 52
 - 7.1.1 Audit Generation..... 52
 - 7.1.2 Audit Storage 52
 - 7.2 Cryptographic Support..... 53
 - 7.2.1 Cryptographic Key Generation..... 53
 - 7.2.2 Cryptographic Operations..... 54
 - 7.2.3 HTTPs Protocol 56
 - 7.2.4 TLS Client Protocol 56
 - 7.2.5 TLS Server Protocol 57
 - 7.3 Identification and Authentication..... 58
 - 7.3.1 Password Management 58
 - 7.3.2 User Identification and Authentication 58
 - 7.3.3 Protected Authentication Feedback 58
 - 7.3.4 X.509 Certificate Validation 58
 - 7.3.5 X.509 Certificate Authentication..... 59
 - 7.3.6 X.509 Certificate Requests 59
 - 7.4 Security Management..... 60
 - 7.5 Protection of the TSF 60
 - 7.5.1 Protection of Administrator Passwords..... 60
 - 7.5.2 Protection of TSF Data (for reading of all symmetric keys) 61
 - 7.5.3 TSF Testing 61
 - 7.5.4 Trusted Update 61
 - 7.5.5 Reliable Time Stamps..... 62
 - 7.6 TOE Access 62
 - 7.7 Trusted Path/Channels 62
 - 7.7.1 Inter-TSF Trusted Channel 62
 - 7.7.2 Trusted Path..... 62
- 8. Terms and Definitions..... 63
- 9. References 65
- Annex A Algorithm Validation Requirements 66

Table 1: VMware Host Details.....	9
Table 2: Threats	12
Table 3: Threats	16
Table 4: Organizational Security Policies	17
Table 5: Assumptions	17
Table 6: Security Objectives for the Operational Environment.....	19
Table 7: Security Functional Requirements	21
Table 8: Auditable Events	23
Table 9: Assurance Requirements	45
Table 10: Conformance Claims	45
Table 11: Cryptographic Algorithms	54
Table 12: Hash Usage.....	55
Table 13: HMAC Usage.....	55
Table 14: TOE Abbreviations and Acronyms.....	63
Table 15: CC Abbreviations and Acronyms	64
Table 16: TOE Guidance Documentation.....	65
Table 17: Common Criteria v3.1 References	65
Table 18: Supporting Documentation.....	65

1. Security Target (ST) Introduction

- The ST introduction shall contain an ST reference, a TOE reference, a TOE overview and a TOE description.
- The ST reference shall uniquely identify the ST.
- The TOE reference shall identify the TOE.

The structure of this document is defined by CC v3.1r4 Part 1 Annex A.2, “Mandatory contents of an ST”:

- Section 1 contains the ST Introduction, including the ST reference, Target of Evaluation (TOE) reference, TOE overview, and TOE description.
- Section 2 contains conformance claims to the Common Criteria (CC) version, Protection Profile (PP) and package claims, as well as rationale for these conformance claims.
- Section 3 contains the security problem definition, which includes threats, Organizational Security Policies (OSP), and assumptions that must be countered, enforced, and upheld by the TOE and its operational environment.
- Section 4 contains statements of security objectives for the TOE, and the TOE operational environment as well as rationale for these security objectives.
- Section 5 contains definitions of any extended security requirements claimed in the ST.
- Section 6 contains the security function requirements (SFR), the security assurance requirements (SAR), as well as the rationale for the claimed SFR and SAR.
- Section 7 contains the TOE summary specification, which includes the detailed specification of the IT security functions

1.1 Security Target Reference

The Security Target reference shall uniquely identify the Security Target.

ST Title: Pulse Secure Virtual Appliance 8.2/5.3 Security Target
ST Version: 3.1
ST Author(s): Acumen Security
ST Publication Date: April 2018
Keywords: Network Virtual Appliance

1.2 Target of Evaluation Reference

The Target of Evaluation reference shall identify the Target of Evaluation.

TOE Developer: Pulse Secure, LLC
2700 Zanker Road
Suite 200
San Jose, CA 95134
TOE Name: Pulse Secure Virtual Appliance 8.2/5.3
TOE Hardware: Dell Power Edge R430/R530 w/Intel Xeon E5-2620v4.

TOE Software: Pulse Connect Secure (PCS) 8.2 and Pulse Policy Secure (PPS) 5.3 running on VMware ESXi 6.0

1.3 Target of Evaluation Overview

1.3.1 TOE Product Type

The TOE is classified as a virtualized network device (a Virtual Appliance that can be connected to a network). The Virtual Appliance consists of Pulse Connect Secure (PCS) 8.2 and Pulse Policy Secure (PPS) 5.3. The appliance's software is built on IVE OS 2.0. The TOE consists of the Virtual Appliance, the VM hypervisor and the hardware platform all of which are delivered with the TOE. Thus, the TOE is considered to be a network device as defined in NDcPP v1.0 modified by TDs #0096 and #0023.

1.3.2 TOE Usage

The TOE is a Virtual Appliance that provides secure remote management, auditing, and updating capabilities. The TOE provides secure remote management using a HTTPS/TLS web interface. The TOE generates audit logs and transmits the audit logs to a remote syslog server over a mutually authenticated TLS channel. The TOE verifies the authenticity of software updates by verifying the digital signature prior to installing any update. The TOE software runs as a virtual appliance on the listed hardware platform.

PCS and PPS are different licenses of the same basic product. PPS supports additional features such as the option for Layer 2 or Layer 3 VPN connectivity, automated patch assessment and remediation, coordinated threat control and identity-enabled firewalls. Both products have the same secure network functionality and all of the functionality differences between them are either unevaluated or excluded functionality.

The scope of the evaluated functionality includes the following,

- Secure remote administration of the TOE via TLS
- Secure Local administration of the TOE
- Secure connectivity with remote audit servers
- Secure access to the management functionality of the TOE
- Identification and authentication of the administrator of the TOE

No other functionality is included within the scope of this evaluation

1.3.3 TOE Major Security Features Summary

- Audit
- Cryptography
- Identification and Authentication
- Security Management
- Protection of the TSF
- TOE Access
- Trusted Path/Channels

1.3.4 TOE IT environment hardware/software/firmware requirements

The TOE's operational environment must provide the following services to support the secure operation of the TOE:

- Syslog Server

- Conformant with RFC 5424 (Syslog Protocol)
- Supporting Syslog over TLS (RFC 5425)
- Acting as a TLSv1.1 and/or TLSv1.2 server
- Supporting Client Certificate authentication
- Supporting at least one of the following cipher suites:
 - TLS_RSA_WITH_AES_128_CBC_SHA
 - TLS_RSA_WITH_AES_256_CBC_SHA
 - TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
 - TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
 - TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
 - TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
 - TLS_RSA_WITH_AES_128_CBC_SHA256
 - TLS_RSA_WITH_AES_256_CBC_SHA256
 - TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
 - TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
 - TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
 - TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
 - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
 - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- Web Browser
 - Internet Explorer 11, Google Chrome 50, or Firefox 38
 - Supporting TLSv1.1 and/or TLSv1.2
 - Supporting Client Certificate authentication
 - Supporting at least one of the following ciphersuites:
 - TLS_RSA_WITH_AES_128_CBC_SHA
 - TLS_RSA_WITH_AES_256_CBC_SHA
 - TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
 - TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
 - TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
 - TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
 - TLS_RSA_WITH_AES_128_CBC_SHA256
 - TLS_RSA_WITH_AES_256_CBC_SHA256
 - TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
 - TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
 - TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
 - TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
 - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
 - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- CRL Server
 - CRL Server conformant with RFC 5280
- Pulse One v2.0 management server (optional)
- DNS Server
 - Conformant with RFC 1035

1.4 Target of Evaluation Description

1.4.1 Target of Evaluation Physical Boundaries

The TOE is a virtual appliance on VMware ESXi 6.0, with a Dell PowerEdge R430/R530 as the hardware platform. ESXi is a bare-metal hypervisor so there is no underlying operating system. In the evaluated configuration, there are no other guest VMs on the physical platform providing non-network device functionality. The TOE software is Pulse Connect Secure v8.2 and Pulse Policy Secure v5.3.

The PCS/PPS software and the platform described below comprise the TOE.

Table 1: VMware Host Details		
Model	Processor	Hypervisor
Power Edge R430/530	Intel Xeon E5-2620 v4	VMware ESXi 6.0

Pulse Connect Secure 8.2 and Pulse Policy Secure v5.3 run on top of and includes IVE OS 2.0, see Figure 1. IVE OS 2.0 is the underlying collection of Kernel/Libraries/Cryptographic Module components that comprises the OE of the cryptographic algorithms.

The guidance documentation that is part of the TOE is listed in Section 9 “References” within Table 16: TOE Guidance Documentation.

1.4.2 Target of Evaluation Logical Boundaries

The logical boundary of the TOE includes those security functions implemented exclusively by the TOE. These security functions are summarized in Section 1.3.3 above and are further described in the following subsections. A more detailed description of the implementation of these security functions are provided in Section 7 “TOE Summary Specification”.

1.4.2.1 Audit

The TOE generates audit records for security relevant events. The TOE maintains a local audit log as well as sending the audit records to a remote Syslog server. Audit records sent to the remote server are protected by a TLS connection. Each audit record includes identity (username, IP address, or process), date and time of the event, type of event, and the outcome of the event. The TOE prevents modification to the local audit log.

1.4.2.2 Cryptographic Operations

The TOE implements CAVP validated cryptographic algorithms for random bit generation, encryption/decryption, authentication, and integrity protection/verification. These algorithms are used to provide security for the TLS and HTTPs connections as well as verifying firmware updates.

1.4.2.3 Identification and Authentication

The TOE authenticates administrative users using a username/password or username/X.509 certificate combination. The TOE does not allow access to any administrative functions prior to successful authentication.

The TOE supports passwords consisting of alphanumeric and special characters and enforces minimum password lengths. The TSF supports and certificates using RSA or ECDSA signature algorithms.

The TOE allows only users to view the login warning banner and send/receive ICMP packets prior to authentication.

1.4.2.4 Security Management

The TOE allows users with the Security Administrator role to administer the TOE over a remote web UI or a local CLI. These interfaces do not allow the Security Administrator to execute arbitrary commands or executables on the TOE.

The TOE can also receive configuration updates from a Pulse One management server.

1.4.2.5 Protection of the TSF

The TOE implements a number of self-protection mechanisms. It does not provide an interface for the reading of secret or private keys. The TOE ensures timestamps, timeouts, and certificate checks are accurate by maintaining a real-time clock as well as polling an NTP server to minimize drift. Upon startup, the TOE runs a suite of self-tests to verify that it is operating correctly. The TOE also verifies the integrity and authenticity of firmware updates by verifying a digital signature of the update prior to installing it.

1.4.2.6 TOE Access

The TOE can be configured to display a warning and consent banner when an administrator attempts to establish an interactive session over the local CLI or remote web UI. The TOE also enforces a configurable inactivity timeout for remote and local administrative sessions.

1.4.2.7 Trusted Path/Channels

The TOE uses TLS to provide a trusted communication channel between itself and remote Syslog servers. The trusted channels utilize X.509 certificates to perform mutual authentication. The TOE initiates the TLS trusted channel with the remote server.

The TOE uses HTTPS/TLS to provide a trusted path between itself and remote administrative users. The TOE does not implement any additional methods of remote administration. The remote administrative users are responsible for initiating the trusted path when they wish to communicate with the TOE.

1.4.2.8 Unevaluated Functionality

The TOE includes the following functionality that is not covered this Security Target and the associated evaluation:

- Layer 3 SSL VPN
- Application VPN
- Endpoint Integrity and Assessment
- Layer 7 Web single sign-on (SSO) via SAML
- Mobile Device Management Integration
- Network Security and Application Access Control Integration
- Federation
- Guest Access
- Anti-Malware Protection and Patch Assessment
- Firewall Listening Service

These features may be used in the evaluated configuration; however, no assurance as to the correct operation of these features is provided.

1.4.2.9 Excluded Functionality

The TOE includes the following functionality that may not be enabled or used in in the CC evaluated configuration:

- DMI Agent
- SNMP Traps
- External Authentication Servers for administrator authentication

1.5 Notation, formatting, and conventions

The notation, formatting, and conventions used in this Security Target are defined below; these styles and clarifying information conventions were developed to aid the reader.

Where necessary, the ST author has added application notes to provide the reader with additional details to aid understanding; they are italicized and usually appear following the element needing clarification. Those notes specific to the TOE are marked “TOE Application Note;” those taken from the collaborative Protection Profile for Network Devices are marked “Application Note <#>.”

The notation conventions that refer to iterations, assignments, selections, and refinements made in this Security Target are in reference to SARs and SFRs taken directly from CC Part 2 and Part 3 as well as any SFRs and SARs taken from a Protection Profile.

The notation used in those PP to indicate iterations, assignments, selections, and refinements of SARs and SFRs taken from CC Part 2 and Part 3 is not carried forward into this document. Additionally, obvious errors in the PP are corrected and noted as such.

The CC permits four component operations (assignment, iteration, refinement, and selection) to be performed on requirement components. These operations are defined in Common Criteria, Part 1; paragraph 6.4.1.3.2, “Permitted operations on components” as:

- Iteration: allows a component to be used more than once with varying operations;
- Assignment: allows the specification of parameters;
- Selection: allows the specification of one or more items from a list; and
- Refinement: allows the addition of details.

Iterations are indicated by a number in parenthesis following the requirement number, e.g., FIA_UAU.1.1(1); the iterated requirement titles are similarly indicated, e.g., FIA_UAU.1(1).

Assignments made by the ST author are identified with **bold text**.

Selections are identified with underlined text.

Refinements that add text use ***bold and italicized text*** to identified the added text. Refinements that performs a deletion, identifies the deleted text with ~~***strikeout, bold, and italicized text***~~.

2. Conformance Claims

2.1 Common Criteria Conformance Claims

This Security Target is conformant to the Common Criteria Version 3.1r4, CC Part 2 extended [C2], and CC Part 3 extended [C3].

2.2 Conformance to Protection Profiles

This Security Target claims exact compliance to the collaborative Protection Profile for Network Devices, Version 1.0, dated February 27, 2015 [NDcPP] and Supporting Document Mandatory Technical Document Evaluation Activities for Network Device cPP, Version 1.0, dated February 27, 2015 [SD]. This Protection Profile will be referred to as cPP or PP for convenience throughout this Security Target.

In addition, the following NIAP Technical Decisions (TD) are applicable to the evaluation (a brief description regarding of the applicability of each TD is provided):

Table 2: Threats	
Description	Rationale
0090 – NIT Technical Decision for FMT_SMF.1.1 Requirement in NDcPP	Applied
0093 – NIT Technical Decision for FIA_X509_EXT.1.1 Requirement in NDcPP	Not applicable – Requirement has been archived
0094 – NIT Technical Decision for validating a published hash in NDcPP	Applied
0095 – NIT Technical Interpretations regarding audit, random bit generation, and entropy in NDcPP	Applied
0096 – NIT Technical Interpretation regarding Virtualization	Applied
0111 – NIT Technical Decision for third party libraries and FCS_CKM.1 in NDcPP and FWcPP	Applied
0112 – NIT Technical Decision for TLS testing in the NDcPP v1.0 and FW cPP v1.0.	Applied
0113 – NIT Technical Decision for testing and trusted updates in the NDcPP v1.0 and FW cPP v1.0	Applied
0114 – NIT Technical Decision for Re-Use of FIPS test results in NDcPP and FWcPP	Applied
0115 – NIT Technical Decision for Transport mode and tunnel mode in IPsec communication in NDcPP and FWcPP	Not applicable – IPsec is not implemented within the TOE
0116 – NIT Technical Decision for a Typo in reference to RSASSA-PKCS1v1_5 in NDcPP and FWcPP	Applied
0117 – NIT Technical Decision for FIA_X509_EXT.1.1 Requirement in NDcPP	Applied
0125 – NIT Technical Decision for Checking validity of peer certificates for HTTPS servers	Applied
0126 – NIT Technical Decision for TLS Mutual Authentication	Applied
0130 – NIT Technical Decision for Requirements for Destruction of Cryptographic Keys	Applied
0143 – NIT Technical Decision for Failure testing for TLS session establishment in NDcPP and FWcPP	Applied

0150 – NIT Technical Decision for Removal of SSH re-key audit events in the NDcPP v1.0 and FW cPP v1.0	Not applicable – SSH is not implemented within the TOE
0151 – NIT Technical Decision for FCS TLSS EXT Testing - Issue 1 in NDcPP v1.0.	Applied
0152 – NIT Technical Decision for Reference identifiers for TLS in the NDcPP v1.0 and FW cPP v1.0	Applied
0153 – NIT Technical Decision for Auditing of NTP Time Changes in the NDcPP v1.0 and FW cPP v1.0	Applied
0154 – NIT Technical Decision for Versions of TOE Software in the NDcPP v1.0 and FW cPP v1.0	Applied
0155 – NIT Technical Decision for TLSS tests using ECDHE in the NDcPP v1.0.	Applied
0156 – NIT Technical Decision for SSL/TLS Version Testing in the NDcPP v1.0 and FW cPP v1.0	Applied
0160 – NIT Technical Decision for Transport mode and tunnel mode in IPSEC communications	Not applicable – IPsec is not implemented within the TOE
0164 – NIT Technical Decision for Negative testing for additional ciphers for SSH	Not applicable – SSH is not implemented within the TOE
0165 – NIT Technical Decision for Sending the ServerKeyExchange message when using RSA	Applied
0167 – NIT Technical Decision for Testing SSH 2^28 packets	Not applicable – SSH is not implemented within the TOE
0168 – NIT Technical Decision for Mandatory requirement for CSR generation	Applied
0169 – NIT Technical Decision for Compliance to RFC5759 and RFC5280 for using CRLs	Applied
0170 – NIT Technical Decision for SNMPv3 Support	Not applicable as the TOE does not claim SNMP functionality
0181 – NIT Technical Decision for Self-testing of integrity of firmware and software.	Applied
0182 – NIT Technical Decision for Handling of X.509 certificates related to ssh-rsa and remote comms.	Not applicable – SSH is not implemented within the TOE
0183 – NIT Technical Decision for Use of the Supporting Document	Applied
0184 – NIT Technical Decision for Mandatory use of X.509 certificates	Applied
0185 – NIT Technical Decision for Channel for Secure Update.	Applied
0186 – NIT Technical Decision for Applicability of X.509 certificate testing to IPsec	Not applicable – IPsec is not implemented within the TOE
0187 – NIT Technical Decision for Clarifying FIA X509 EXT.1 test 1	Applied
0188 – NIT Technical Decision for Optional use of X.509 certificates for digital signatures	Applied
0189 – NIT Technical Decision for SSH Server Encryption Algorithms	Not applicable – SSH is not implemented within the TOE
0191 – NIT Technical Decision for Using secp521r1 for TLS communication	Applied
0195 – NIT Technical Decision Making DH Group 14 optional in FCS IPSEC EXT.1.11	Not applicable – IPsec is not implemented within the TOE

0199 – NIT Technical Decision for Elliptic Curves for Signatures	Applied
0200 – NIT Technical Decision for Password authentication for SSH clients	Not applicable – SSH is not implemented within the TOE
0201 – NIT Technical Decision for Use of intermediate CA certificates and certificate hierarchy depth	Applied
0209 – Additional DH Group added as selection for IKE Protocols	Not applicable – IPsec is not implemented within the TOE
0223 – NIT Technical Decision for "Expected" vs "unexpected" DN's for IPsec Communications	Not applicable – IPsec is not implemented within the TOE
0224 – NIT Technical Decision Making DH Group 14 optional in FCS_IPSEC_EXT.1.11	Not applicable – IPsec is not implemented within the TOE
0225 – NIT Technical Decision for Make CBC cipher suites optional in IPsec	Not applicable – IPsec is not implemented within the TOE
0226 – NIT Technical Decision for TLS Encryption Algorithms	Applied
0227 – NIT Technical Decision for TOE acting as a TLS Client and RSA key generation	Applied
0228 – NIT Technical Decision for CA certificates - basicConstraints validation	Applied
0235 – NIT Technical Decision adding DH group 14 to the selection in FCS_CKM.2	Applied
0255 – NIT Technical Decision for TLS Server Tests - Issue 3: Verification of application of encryption	Applied
0256 – NIT Technical Decision for Handling of TLS connections with and without mutual authentication	Applied
0257 – NIT Technical Decision for Updating FCS_DTLSC_EXT.x.2/FCS_TLSC_EXT.x.2 Tests 1-4	Applied
0262 – NIT Technical Decision for TLS server testing - Empty Certificate Authorities list	Applied

2.3 Conformance to Security Packages

This Security Target does not claim conformance to any security function requirements or security assurance requirements packages, neither as package-conformant or package-augmented.

2.4 Conformance Claims Rationale

To demonstrate that exact conformance is met, this rationale shows all threats are addressed, all OSP are satisfied, no additional assumptions are made, all objectives have been addressed, and all SFRs and SARs have been instantiated.

The following address the completeness of the threats, OSP, and objectives, limitations on the assumptions, and instantiation of the SFRs and SARs:

- Threats
 - All threats defined in the cPP are carried forward to this ST;
 - No additional threats have been defined in this ST.
- Organizational Security Policies
 - All OSP defined in the cPP are carried forward to this ST;

- No additional OSPs have been defined in this ST.
- Assumptions
 - All assumptions defined in the cPP are carried forward to this ST;
 - No additional assumptions for the operational environment have been defined in this ST.
- Objectives
 - All objectives defined in the cPP are carried forward to this ST.
- All SFRs and SARs defined in the cPP are carried forward to this Security Target.

Rationale presented in the body of this ST shows all assumptions on the operational environment have been upheld, all the OSP are enforced, all defined objectives have been met and these objectives counter the defined threats.

Additionally, all SFRs and SARs defined in the cPP have been properly instantiated in this Security Target; therefore, this ST shows exact compliance to the cPP.

3. Security Problem Definition

3.1 Threats

The following table defines the security threats for the TOE, characterized by a threat agent, an asset, and an adverse action of that threat agent on that asset. These threats are taken directly from the PP unchanged.

Table 3: Threats	
Threat	Description
T.UNAUTHORIZED_ADMINISTRATOR_ACCESS	Threat agents may attempt to gain administrator access to the network device by nefarious means such as masquerading as an administrator to the device, masquerading as the device to an administrator, replaying an administrative session (in its entirety, or selected portions), or performing man-in-the-middle attacks, which would provide access to the administrative session, or sessions between network devices. Successfully gaining administrator access allows malicious actions that compromise the security functionality of the device and the network on which it resides.
T.WEAK_CRYPTOGRAPHY	Threat agents may exploit weak cryptographic algorithms or perform a cryptographic exhaust against the key space. Poorly chosen encryption algorithms, modes, and key sizes will allow attackers to compromise the algorithms, or brute force exhaust the key space and give them unauthorized access allowing them to read, manipulate and/or control the traffic with minimal effort.
T.UNTRUSTED_COMMUNICATION_CHANNELS	Threat agents may attempt to target network devices that do not use standardized secure tunneling protocols to protect the critical network traffic. Attackers may take advantage of poorly designed protocols or poor key management to successfully perform man-in-the-middle attacks, replay attacks, etc. Successful attacks will result in loss of confidentiality and integrity of the critical network traffic, and potentially could lead to a compromise of the network device itself.
T.WEAK_AUTHENTICATION_ENDPOINTS	Threat agents may take advantage of secure protocols that use weak methods to authenticate the endpoints – e.g., shared password that is guessable or transported as plaintext. The consequences are the same as a poorly designed protocol, the attacker could masquerade as the administrator or another device, and the attacker could insert themselves into the network stream and perform a man-in-the-middle attack. The result is the critical network traffic is exposed and there could be a loss of confidentiality and integrity, and potentially the network device itself could be compromised.
T.UPDATE_COMPROMISE	Threat agents may attempt to provide a compromised update of the software or firmware which undermines the security functionality of the device. Non-validated updates or updates validated using non-secure or weak cryptography leave the update firmware vulnerable to surreptitious alteration.
T.UNDETECTED_ACTIVITY	Threat agents may attempt to access, change, and/or modify the security functionality of the network device without administrator awareness. This could result in the attacker finding an avenue (e.g., misconfiguration, flaw in the product) to compromise the device and the administrator would have no knowledge that the device has been compromised.
T.SECURITY_FUNCTIONALITY_COMPROMISE	Threat agents may compromise credentials and device data enabling continued access to the network device and its critical data. The compromise of credentials include replacing existing credentials with an attacker's credentials, modifying

Table 3: Threats	
Threat	Description
	existing credentials, or obtaining the administrator or device credentials for use by the attacker.
T.PASSWORD_CRACKING	Threat agents may be able to take advantage of weak administrative passwords to gain privileged access to the device. Having privileged access to the device provides the attacker unfettered access to the network traffic, and may allow them to take advantage of any trust relationships with other network devices.
T.SECURITY_FUNCTIONALITY_FAILURE	A component of the network device may fail during start-up or during operations causing a compromise or failure in the security functionality of the network device, leaving the device susceptible to attackers.

3.2 Organizational Security Policies

The following table defines the organizational security policies (OSPs) which are a set of rules, practices, and procedures imposed by an organization to address its security needs. These OSPs are taken directly from the PP unchanged.

Table 4: Organizational Security Policies	
OSP	Description
P.ACCESS_BANNER	The TOE shall display an initial banner describing restrictions of use, legal agreements, or any other appropriate information to which users consent by accessing the TOE.

3.3 Assumptions

This section describes the assumptions on the operational environment in which the TOE is intended to be used. It includes information about the physical, personnel, and connectivity aspects of the environment. The operational environment must be managed in accordance with the provided guidance documentation. The following table defines specific conditions that are assumed to exist in an environment where the TOE is deployed. These assumptions are taken directly from the PP unchanged.

Table 5: Assumptions	
Assumption	Description
A.PHYSICAL_PROTECTION	The network device is assumed to be physically protected in its operational environment and not subject to physical attacks that compromise the security and/or interfere with the device's physical interconnections and correct operation. This protection is assumed to be sufficient to protect the device and the data it contains. As a result, the cPP will not include any requirements on physical tamper protection or other physical attack mitigations. The cPP will not expect the product to defend against physical access to the device that allows unauthorized entities to extract data, bypass other controls, or otherwise manipulate the device.
A.LIMITED_FUNCTIONALITY	The device is assumed to provide networking functionality as its core function and not provide functionality/services that could be deemed as general purpose computing. For example the device should not provide computing platform for general purpose Applications (unrelated to networking functionality).
A.NO_THRU_TRAFFIC_PROTECTION	A standard/generic network device does not provide any assurance regarding the protection of traffic that traverses it. The intent is for the network device to protect data that originates on or is destined to the device itself, to include administrative data and audit data. Traffic that is traversing the network device, destined for another network entity, is not

Table 5: Assumptions	
Assumption	Description
	covered by the ND cPP. It is assumed that this protection will be covered by cPPs for particular types of network devices (e.g, firewall).
A.TRUSTED_ADMINISTRATOR	The Security Administrator(s) for the network device are assumed to be trusted and to act in the best interest of security for the organization. This includes being appropriately trained, following policy, and adhering to guidance documentation. Administrators are trusted to ensure passwords/credentials have sufficient strength and entropy and to lack malicious intent when administering the device. The network device is not expected to be capable of defending against a malicious administrator that actively works to bypass or compromise the security of the device.
A.REGULAR_UPDATES	The network device firmware and software is assumed to be updated by an administrator on a regular basis in response to the release of product updates due to known vulnerabilities.
A.ADMIN_CREDENTIALS_SECURE	The administrator's credentials (private key) used to access the network device are protected by the platform on which they reside.

4. Security Objectives

4.1 Security Objectives for the Operational Environment

Table 6: Security Objectives for the Operational Environment	
Objective	Description
OE.PHYSICAL	Physical security, commensurate with the value of the TOE and the data it contains, is provided by the environment.
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.NO_THRU_TRAFFIC_PROTECTION	The TOE does not provide any protection of traffic that traverses it. It is assumed that protection of this traffic will be covered by other security and assurance measures in the operational environment
OE.TRUSTED_ADMIN	TOE Administrators are trusted to follow and apply all guidance documentation in a trusted manner.
OE.UPDATE	The TOE firmware and software is updated by an administrator on a regular basis in response to the release of product updates due to known vulnerabilities.
OE.ADMIN_CREDENTIALS_SECURE	The administrator's credentials (private key) used to access the TOE must be protected on any other platform on which they reside.

5. Extended Components Definition

This section provides definition of the extended security functional and assurance requirements; the components that are CC Part 2 extended, and CC Part 3 extended, i.e., NIAP interpreted requirements, and extended requirements.

5.1 Extended Security Functional Requirements Definitions

There are no extended Security Functional Requirements defined in this Security Target. All extended SFRs are defined in the [NDcPP].

5.2 Extended Security Assurance Requirement Definitions

There are no extended Security Assurance Requirements defined in this Security Target. All extended SFRs are defined in the [NDcPP] and [SD].

6. Security Requirements

This section describes the security functional and assurance requirements for the TOE; those that are CC Part 2 conformant, CC Part 2 extended, CC Part 3 conformant, and CC Part 3 extended.

6.1 Security Function Requirements

This section describes the functional requirements for the TOE. The security functional requirement components in this security target are CC Part 2 conformant or CC Part 2 extended as defined in Section 2, Conformance Claims. Operations that were performed in the cPP are not signified in this section. Operations performed by the ST are denoted according to the formatting conventions in Section 1.5.

Table 7: Security Functional Requirements	
SFR	Description
FAU_GEN.1	Audit Data Generation
FAU_GEN.2	User Audit Association
FAU_STG.1	Protected Audit Trail Storage
FAU_STG_EXT.1	Protected Audit Event Storage
FAU_STG_EXT.3	Display Warning for Local Storage Space
FCS_CKM.1	Cryptographic Key Generation (Refined)
FCS_CKM.2	Cryptographic Key Establishment (Refined)
FCS_CKM.4	Cryptographic Key Destruction
FCS_COP.1(1)	Cryptographic Operation (AES Data Encryption/Decryption)
FCS_COP.1(2)	Cryptographic Operation (Signature Generation and Verification)
FCS_COP.1(3)	Cryptographic Operation (Hash Algorithm)
FCS_COP.1(4)	Cryptographic Operation (Keyed Hash Algorithm)
FCS_HTTPS_EXT.1	HTTPS Protocol
FCS_RBG_EXT.1	Random Bit Generation
FCS_TLSC_EXT.1	TLS Client Protocol
FCS_TLSC_EXT.2	TLS Client Protocol with Authentication
FCS_TLSS_EXT.1	TLS Server Protocol
FIA_PMG_EXT.1	Password Management
FIA_UIA_EXT.1	User Identification and Authentication
FIA_UAU_EXT.2	Password-based Authentication Mechanism
FIA_UAU.7	Protected Authentication Feedback
FIA_X509_EXT.1	X.509 Certificate Validation
FIA_X509_EXT.2	X.509 Certificate Authentication
FIA_X509_EXT.3	X.509 Certificate Requests
FMT_MOF.1(1)/Trusted Update	Management of Security Functions Behaviour
FMT_MOF.1(1)/Audit	Management of Security Functions Behaviour
FMT_MOF.1(2)/Audit	Management of Security Functions Behaviour

Table 7: Security Functional Requirements	
SFR	Description
FMT_MOF.1(1)/AdminAct	Management of Security Functions Behaviour
FMT_MTD.1	Management of TSF Data
FMT_MTD.1/AdminAct	Management of TSF Data
FMT_SMF.1	Specification of Management Functions
FMT_SMR.2	Restrictions on Security Roles
FPT_SKP_EXT.1	Protection of TSF Data (for reading all symmetric keys)
FPT_APW_EXT.1	Protection of Administrator Passwords
FPT_TST_EXT.1	TSF Testing
FPT_TUD_EXT.1	Trusted Update
FPT_STM.1	Reliable Time Stamps
FTA_SSL_EXT.1	TSF-initiated Session Locking
FTA_SSL.3	TSF-initiated Termination
FTA_SSL.4	User-initiated Termination
FTA_TAB.1	Default TOE Access Banners
FPT_ITC.1	Inter-TSF Trusted Channel
FPT_TRP.1	Trusted Path

6.1.1 Class FAU: Security Audit

6.1.1.1 FAU_GEN.1 Audit Data Generation

FAU_GEN.1.1

The TSF shall be able to generate an audit record of the following auditable events:

- a) Start-up and shut-down of the audit functions;
- b) All auditable events for the not specified level of audit; and
- c) All administrative actions comprising:
 - Administrative login and logout (name of user account shall be logged if individual user accounts are required for administrators).
 - Security related configuration changes (in addition to the information that a change occurred it shall be logged what has been changed).
 - Generating/import of, changing, or deleting of cryptographic keys (in addition to the action itself a unique key name or key reference shall be logged).
 - Resetting passwords (name of related user account shall be logged).
 - Starting and stopping services (if applicable)
 - no other actions;
- d) Specifically defined auditable events listed in Table 8.

Application Note 1

If the list of 'administrative actions' appears to be incomplete, the assignment in the selection should be used to list additional administrative actions which are audited.

The ST author replaces the cross-reference to the table of audit events with an appropriate cross-reference for the ST. This must also include the relevant parts of Table 3 and Table 4 for optional and selection-based SFRs included in the ST.

Application Note 2

The ST author can include other auditable events directly in the table; they are not limited to the list presented.

The TSS should identify what information is logged to identify the relevant key for the administrative task of generating/import of, changing, or deleting of cryptographic keys.

With respect to FAU_GEN.1.1 the term ‘services’ refers to trusted path and trusted channel communications, on demand self-tests, trusted update and administrator sessions (that exist under the trusted path) (e.g. netconf).

FAU_GEN1.2

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

- a) Date and time of the event, type of event, subject identity, and the outcome (success or failure) of the event; and
- b) 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 8.

Application Note 3

The ST author replaces the cross-reference to the table of audit events with an appropriate cross-reference for the ST. This must also include the relevant parts of cPP Table 3 and cPP Table 4 for optional and selection-based SFRs included in the ST.

Table 8: Auditable Events		
SFR	Auditable Events	Additional Audit Record Contents
FAU_GEN.1	None.	None.
FAU_GEN.2	None.	None.
FAU_STG.1	None.	None.
FAU_STG_EXT.1	None.	None.
FAU_STG_EXT.3	Warning about low storage space for audit events	None.
FCS_CKM.1	None.	None.
FCS_CKM.2	None.	None.
FCS_CKM.4	None.	None.
FCS_COP.1(1)	None.	None.
FCS_COP.1(2)	None.	None.
FCS_COP.1(3)	None.	None.
FCS_COP.1(4)	None.	None.
FCS_HTTPS_EXT.1	Failure to establish a HTTPS session	Reason for failure
FCS_RBG_EXT.1	None.	None.
FCS_TLSC_EXT.1	Failure to establish a TLS Session	Reason for failure

Table 8: Auditable Events		
SFR	Auditable Events	Additional Audit Record Contents
FCS_TLSC_EXT.2	Failure to establish a TLS Session	Reason for failure
FCS_TLSS_EXT.1	Failure to establish a TLS Session	Reason for failure
FIA_PMG_EXT.1	None.	None.
FIA_UIA_EXT.1	All use of the identification and authentication mechanism.	Provided user identity, origin of the attempt (e.g., IP address).
FIA_UAU_EXT.2	All use of the identification and authentication mechanism.	Origin of the attempt (e.g., IP address).
FIA_UAU.7	None.	None.
FIA_X509_EXT.1	Unsuccessful attempt to validate a certificate.	Reason for Failure
FIA_X509_EXT.2	None.	None.
FIA_X509_EXT.3	None.	None.
FMT_MOF.1(1)/TrustedUpdate	Any attempt to initiate a manual update.	None.
FMT_MOF.1(1)/Audit	Modification of the behaviour of the transmission of audit data to an external identity.	None.
FMT_MOF.1(2)/Audit	Modification of the behaviour of the handling of audit data.	None.
FMT_MOF.1(1)/AdminAct	Modification of the behavior of the TSF.	None.
FMT_MTD.1	All management activities of TSF data.	None.
FMT_MTD.1/AdminAct	Modification, deletion, generation/import of cryptographic keys.	None.
FMT_SMF.1	None.	None.
FMT_SMR.2	None.	None.
FPT_SKP_EXT.1	None.	None.
FPT_APW_EXT.1	None.	None.
FPT_TST_EXT.1	None.	None.
FPT_TUD_EXT.1	Initiation of update; result of the update attempt (success or failure)	No additional information.
FPT_STM.1	Changes to time.	The old and new values for the time. Origin of the attempt to change time for success and failure (e.g., IP address).
FTA_SSL_EXT.1	Any attempts at unlocking of an interactive session.	None.
FTA_SSL.3	The termination of a remote session by the session locking mechanism.	None.
FTA_SSL.4	The termination of an interactive session.	None.

Table 8: Auditable Events		
SFR	Auditable Events	Additional Audit Record Contents
FTA_TAB.1	None.	None.
FTP_ITC.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.
FTP_TRP.1	Initiation of the trusted path. Termination of the trusted path. Failure of the trusted path functions.	Identification of the claimed user identity.

Application Note 4

Additional audit events will apply to the TOE depending on the optional and selection-based requirements adopted from Appendix A and Appendix B. The ST author must therefore include the relevant additional events specified in the tables in [NDcPP] Table 3 and [NDcPP] Table 4. The audit event for FIA_X509_EXT.1 is based on the TOE not being able to complete the certificate validation by ensuring the following:

- *the presence of the basicConstraints extension and that the CA flag is set to TRUE for all CA certificates.*
- *Verification of the digital signature of the trusted hierarchical CA*
- *read/access the CRL or access the OCSP server.*

If any of these checks fails, then an audit event with the failure should be written to the audit log

6.1.1.2 FAU_GEN.2 User Identity Association**FAU_GEN.2.1**

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

6.1.1.3 FAU_STG.1 Protected Audit Trail Storage**FAU_STG.1.1**

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

FAU_STG.1.2

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

6.1.1.4 FAU_STG_EXT.1 Protected Audit Event Storage**FAU_STG_EXT.1.1**

The TSF shall be able to transmit the generated audit data to an external IT entity using a trusted channel according FTP_ITC.1.

Application Note 5

For selecting the option of transmission of generated audit data to an external IT entity the TOE relies on a non-TOE audit server for storage and review of audit records. The storage of these audit records and the

ability to allow the administrator to review these audit records is provided by the operational environment in that case.

FAU_STG_EXT.1.2

The TSF shall be able to store generated audit data on the TOE itself.

FAU_STG_EXT.1.3

The TSF shall overwrite previous audit records according to the following rule: the oldest log file is overwritten when the local storage space for audit data is full.

Application Note 6

The external log server might be used as alternative storage space in case the local storage space is full. The 'other action' could in this case be defined as 'send the new audit data to an external IT entity'.

6.1.1.5 FAU_STG_EXT.3 Display Warning for Local Storage Space

FAU_STG_EXT.3.1

The TSF shall generate a warning to inform the user before the local space to store audit data is used up and/or the TOE will lose audit data due to insufficient local space.

Application Note 42

This option should be chosen if the TOE generates as warning to inform the user before the local storage space for audit data is used up. This might be useful if auditable events are stored on local storage space only.

It has to be ensured that the warning message required by FAU_STG_EXT.1.3 can be communicated to the user. The communication should be done via the audit log itself because it cannot be guaranteed that an administrative session is active at the time the event occurs.

6.1.2 Class FCS: Cryptographic Support

6.1.2.1 FCS_CKM.1 Cryptographic Key Generation

FCS_CKM.1.1

The TSF shall generate asymmetric cryptographic keys in accordance with a specified cryptographic 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 that meet the following: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.4;

Application Note 7

The ST author selects all key generation schemes used for key establishment and device authentication. When key generation is used for key establishment, the schemes in FCS_CKM.2.1 and selected cryptographic protocols must match the selection. When key generation is used for device authentication, the public key is expected to be associated with an X.509v3 certificate.

If the TOE acts as a receiver in the RSA key establishment scheme, the TOE does not need to implement RSA key generation.

6.1.2.2 FCS_CKM.2 Cryptographic Key Establishment

FCS_CKM.2.1

The TSF shall perform cryptographic key establishment in accordance with a specified cryptographic key establishment method:

- RSA-based key establishment schemes that meets the following: NIST Special Publication 800-56B, "Recommendation for Pair-Wise Key Establishment Schemes Using Integer Factorization Cryptography";
- Elliptic curve-based key establishment schemes that meets the following: NIST Special Publication 800-56A, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography";

Application Note 8

This is a refinement of the SFR FCS_CKM.2 to deal with key establishment rather than key distribution.

The ST author selects all key establishment schemes used for the selected cryptographic protocols.

The RSA-based key establishment schemes are described in Section 9 of NIST SP 800-56B; however, Section 9 relies on implementation of other sections in SP 800-56B. If the TOE acts as a receiver in the RSA key establishment scheme, the TOE does not need to implement RSA key generation.

The elliptic curves used for the key establishment scheme correlate with the curves specified in FCS_CKM.1.1.

The domain parameters used for the finite field-based key establishment scheme are specified by the key generation according to FCS_CKM.1.1.

6.1.2.3 FCS_CKM.4 Cryptographic Key Destruction

FCS_CKM.4.1

The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method:

- For plaintext keys in volatile storage, the destruction shall be executed by a single overwrite consisting of zeroes.
 -
- For plaintext keys in non-volatile storage, the destruction shall be executed by the invocation of an interface provided by the TSF that logically addresses the storage location of the key and performs a **three-pass** overwrite consisting of a **pseudo-random pattern**

that meets the following: No Standard.

ST Application Note

FCS_CKM.4 was updated as specified by TD0130.

6.1.2.4 FCS_COP.1(1) Cryptographic Operation (AES Data Encryption/Decryption)

FCS_COP.1.1(1)

The TSF shall perform encryption/decryption in accordance with a specified cryptographic algorithm AES used in CBC, GCM mode and cryptographic key sizes 128 bits, 256 bits that meet the following: AES as specified in ISO 18033-3, CBC as specified in ISO 10116, GCM as specified in ISO 19772.

Application Note 9

For the first selection of FCS_COP.1.1(1), the ST author should choose the mode or modes in which AES operates. For the second selection, the ST author should choose the key sizes that are supported by this functionality. The modes and key sizes selected here correspond to the cipher suite selections made in the trusted channel requirements.

6.1.2.5 FCS_COP.1(2) Cryptographic Operation (Signature Generation and Verification)

FCS_COP.1.1(2)

The TSF shall perform cryptographic signature services (generation and verification) in accordance with a specified cryptographic algorithm

- RSA Digital Signature Algorithm and cryptographic key sizes (modulus) 2048 bits, 3072 bits
- Elliptic Curve Digital Signature Algorithm and cryptographic key sizes 256 bits, 384 bits

that meet the following:

- For RSA schemes: FIPS PUB 186-4, “Digital Signature Standard (DSS)”, Section 5.5, using PKCS #1 v2.1 Signature Schemes RSASSA-PSS and/or RSASSA-PKCS1v1_5; ISO/IEC 9796-2, Digital signature scheme 2 or Digital Signature scheme 3,
- For ECDSA schemes: FIPS PUB 186-4, “Digital Signature Standard (DSS)”, Section 6 and Appendix D, Implementing “NIST curves” P-256, P-384; ISO/IEC 14888-3, Section 6.4.

Application Note 10

The ST Author should choose the algorithm implemented to perform digital signatures. For the algorithm(s) chosen, the ST author should make the appropriate assignments/selections to specify the parameters that are implemented for that algorithm.

ST Application Note

FCS_COP.1(2) was updated according to TD0116 and TD0199.

6.1.2.6 FCS_COP.1(3) Cryptographic Operation (Hash Algorithm)

FCS_COP.1.1(3)

The TSF shall perform cryptographic hashing services in accordance with a specified cryptographic algorithm SHA-1, SHA-256, SHA-384, SHA-512 that meet the following: ISO/IEC 10118-3:2004.

Application Note 11

Vendors are strongly encouraged to implement updated protocols that support the SHA-2 family; until updated protocols are supported, this PP allows support for SHA-1 implementations in compliance with SP 800-131A. The hash selection should be consistent with the overall strength of the algorithm used for FCS_COP.1(1) and FCS_COP.1(2) (for example, SHA 256 for 128-bit keys).

6.1.2.7 FCS_COP.1(4) Cryptographic Operation (Keyed Hash Algorithm)

FCS_COP.1.1(4)

The TSF shall perform keyed-hash message authentication in accordance with a specified cryptographic algorithm HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-384 and cryptographic key sizes **128 bits, 160 bits, 192 bits, 256 bits, 352, bits, 384 bits, 1024 bits, 2048 bits** and message digest sizes 160, 256, 384 bits that meet the following: ISO/IEC 9797-2:2011, Section 7 “MAC Algorithm 2”.

Application Note 12

The key size [k] in the assignment falls into a range between L1 and L2 (defined in ISO/IEC 10118 for the appropriate hash function). For example, for SHA-256, L1=512, L2=256, where $L2 \leq k \leq L1$.

6.1.2.8 FCS_HTTPS_EXT.1 HTTPS Protocol

FCS_HTTPS_EXT.1.1

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

Application Note 50

The ST author must provide enough detail to determine how the implementation is complying with the standard(s) identified; this can be done either by adding elements to this component, or by additional detail in the TSS.

FCS_HTTPS_EXT.1.2

The TSF shall implement HTTPS using TLS.

FCS_HTTPS_EXT.1.3

The TSF shall establish the connection only if the peer presents a valid certificate during handshake.

Application Note 51

Select ‘the peer presents a valid certificate’ if the TOE acts as a client, or if mutual certificate-based authentication is enforced when the TOE acts as a client or a server. Certificate validity must be determined according to FIA_X509_EXT.1/Rev if HTTPS is used for FPT_TRP.1/Admin or FTP_ITC.1, and on FIA_X509_EXT.1/ITT if HTTPS is used for FPT_ITT.1.

Select ‘the peer initiates handshake’ if the TOE acts as a server that does not enforce mutual certificate-based authentication. It is understood that in such cases peer authentication is achieved by other means.

ST Application Note

FCS_HTTPS_EXT.1.3 was updated as specified by TD0125.

6.1.2.9 FCS_RBG_EXT.1 Random Bit Generation

FCS_RBG_EXT.1.1

The TSF shall perform all deterministic random bit generation services in accordance with ISO/IEC 18031:2011 using CTR_DRBG (AES).

FCS_RBG_EXT.1.2

The deterministic RBG shall be seeded by at least one entropy source that accumulates entropy from 2 software-based noise source with a minimum of 256 bits of entropy at least equal to the greatest security strength, according to ISO/IEC 18031:2011 Table C.1 “Security Strength Table for Hash Functions”, of the keys and hashes that it will generate.

Application Note 13

For the first selection in FCS_RBG_EXT.1.2, the ST selects at least one of the types of noise sources. If the TOE contains multiple noise sources of the same type, the ST author fills the assignment with the appropriate number for each type of source (e.g., 2 software-based noise sources, 1 hardware-based noise source). The documentation and tests required in the Evaluation Activity for this element necessarily cover each source indicated in the ST.

ISO/IEC 18031:2011 contains three different methods of generating random numbers; each of these, in turn, depends on underlying cryptographic primitives (hash functions/ciphers). The ST author will select the function used, and include the specific underlying cryptographic primitives used in the requirement. While any of the identified hash functions (SHA-1, SHA-224, SHA-256, SHA-384, SHA-512) are allowed for Hash_DRBG or HMAC_DRBG, only AES-based implementations for CTR_DRBG are allowed.

If the key length for the AES implementation used here is different than that used to encrypt the user data, then FCS_COP.1 may have to be adjusted or iterated to reflect the different key length. For the selection in FCS_RBG_EXT.1.2, the ST author selects the minimum number of bits of entropy that is used to seed the RBG.

6.1.2.10 FCS_TLSC_EXT.1 TLS Client Protocol

FCS_TLSC_EXT.1.1

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

- TLS_RSA_WITH_AES_128_CBC_SHA as defined in RFC 3268
- TLS_RSA_WITH_AES_256_CBC_SHA256 as defined in RFC 5246
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289

Application Note 74

The ciphersuites to be tested in the evaluated configuration are limited by this requirement. The ST author should select the optional ciphersuites that are supported; if there are no ciphersuites supported other than the mandatory suites, then “None” should be selected. It is necessary to limit the ciphersuites that can be used in an evaluated configuration administratively on the server in the test environment. TLS_RSA_WITH_AES_128_CBC_SHA is required in order to ensure compliance with RFC 5246.

These requirements will be revisited as new TLS versions are standardized by the IETF.

In a future version of this CPP TLS v1.2 will be required for all TOEs.

FCS_TLSC_EXT.1.2

The TSF shall verify that the presented identifier matches the reference identifier according to RFC 6125.

Application Note 75

The rules for verification of identify are described in Section 6 of RFC 6125. The reference identifier is established by the user (e.g. entering a URL into a web browser or clicking a link), by configuration (e.g. configuring the name of a mail server or authentication server), or by an application (e.g. a parameter of an API) depending on the application service. Based on a singular reference identifier's source domain and application service type (e.g. HTTP, SIP, LDAP), the client establishes all reference identifiers which are acceptable, such as a Common Name for the Subject Name field of the certificate and a (case-insensitive) DNS name, URI name, and Service Name for the Subject Alternative Name field. The client then compares this list of all acceptable reference identifiers to the presented identifiers in the TLS server's certificate.

The preferred method for verification is the Subject Alternative Name using DNS names, URI names, or Service Names. Verification using the Common Name is required for the purposes of backwards compatibility. Additionally, support for use of IP addresses in the Subject Name or Subject Alternative name is discouraged as against best practices but may be implemented. Finally, the client should avoid constructing reference identifiers using wildcards. However, if the presented identifiers include wildcards, the client must follow the best practices regarding matching; these best practices are captured in the assurance activity.

FCS_TLSC_EXT.1.3

The TSF shall only establish a trusted channel if the peer certificate is valid.

Application Note 76

Validity is determined by the identifier verification, certificate path, the expiration date, and the revocation status in accordance with RFC 5280. Certificate validity is tested in accordance with testing performed for FIA_X509_EXT.1.

FCS_TLSC_EXT.1.4

The TSF shall present the Supported Elliptic Curves Extension in the Client Hello with the following NIST curves: secp256r1, secp384r1 and no other curves.

Application Note 77

If ciphersuites with elliptic curves were selected in FCS_TLSC_EXT.1.1, a selection of one or more curves is required. If no ciphersuites with elliptic curves were selected in FCS_TLSC_EXT.1.1, then 'none' should be selected. This requirement limits the elliptic curves allowed for authentication and key agreement to the NIST curves from FCS_COP.1(2) and FCS_CKM.1 and FCS_CKM.2. This extension is required for clients supporting Elliptic Curve ciphersuites.

6.1.2.11 FCS_TLSC_EXT.2 Extended: TLS Client Protocol with Authentication

FCS_TLSC_EXT.2.1

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

- TLS_RSA_WITH_AES_128_CBC_SHA as defined in RFC 3268
- TLS_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_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_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5246
- TLS_RSA_WITH_AES_256_CBC_SHA256 as defined in RFC 5246
- 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
- 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

Application Note 78

The ciphersuites to be tested in the evaluated configuration are limited by this requirement. The ST author should select the optional ciphersuites that are supported; if there are no ciphersuites supported other than the mandatory suites, then “None” should be selected. It is necessary to limit the ciphersuites that can be used in an evaluated configuration administratively on the server in the test environment. The Suite B algorithms listed above (RFC 6460) are the preferred algorithms for implementation. TLS_RSA_WITH_AES_128_CBC_SHA is required in order to ensure compliance with RFC 5246.

These requirements will be revisited as new TLS versions are standardized by the IETF.

In a future version of this cPP TLS v1.2 will be required for all TOEs.

FCS_TLSC_EXT.2.2

The TSF shall verify that the presented identifier matches the reference identifier according to RFC 6125.

Application Note 79

The rules for verification of identify are described in Section 6 of RFC 6125. The reference identifier is established by the user (e.g. entering a URL into a web browser or clicking a link), by configuration (e.g. configuring the name of a mail server or authentication server), or by an application (e.g. a parameter of an API) depending on the application service. Based on a singular reference identifier’s source domain and application service type (e.g. HTTP, SIP, LDAP), the client establishes all reference identifiers which are acceptable, such as a Common Name for the Subject Name field of the certificate and a (case-insensitive) DNS name, URI name, and Service Name for the Subject Alternative Name field. The client then compares this list of all acceptable reference identifiers to the presented identifiers in the TLS server’s certificate. The preferred method for verification is the Subject Alternative Name using DNS names, URI names, or Service Names. Verification using the Common Name is required for the purposes of backwards compatibility. Additionally, support for use of IP addresses in the Subject Name or Subject Alternative name is discouraged as against best practices but may be implemented. Finally, the client should avoid constructing reference identifiers using wildcards. However, if the presented identifiers include wildcards, the client must follow the best practices regarding matching; these best practices are captured in the assurance activity.

FCS_TLSC_EXT.2.3

The TSF shall only establish a trusted channel if the peer certificate is valid.

Application Note 80

Validity is determined by the identifier verification, certificate path, the expiration date, and the revocation status in accordance with RFC 5280. Certificate validity shall be tested in accordance with testing performed for FIA_X509_EXT.1.

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 and no other curves.

Application Note 81

If ciphersuites with elliptic curves were selected in FCS_TLSC_EXT.2.1, a selection of one or more curves is required. If no ciphersuites with elliptic curves were selected in FCS_TLS_EXT.2.1, then 'none' should be selected. This requirement limits the elliptic curves allowed for authentication and key agreement to the NIST curves from FCS_COP.1(2) and FCS_CKM.1 and FCS_CKM.2. This extension is required for clients supporting Elliptic Curve ciphersuites.

FCS_TLSC_EXT.2.5

The TSF shall support mutual authentication using X.509v3 certificates.

Application Note 82

The use of X.509v3 certificates for TLS is addressed in FIA_X509_EXT.2.1. This requirement adds that the client must be capable of presenting a certificate to a TLS server for TLS mutual authentication.

6.1.2.12 FCS_TLSS_EXT.1 TLS Server Protocol

FCS_TLSS_EXT.1.1

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

- TLS_RSA_WITH_AES_128_CBC_SHA as defined in RFC 3268
- TLS_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_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_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5246
- TLS_RSA_WITH_AES_256_CBC_SHA256 as defined in RFC 5246
- 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
- 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.

Application Note 83

The ciphersuites to be tested in the evaluated configuration are limited by this requirement. The ST author should select the optional ciphersuites that are supported; if there are no ciphersuites supported other

than the mandatory suites, then “None” should be selected. It is necessary to limit the ciphersuites that can be used in an evaluated configuration administratively on the server in the test environment. The Suite B algorithms listed above (RFC 6460) are the preferred algorithms for implementation. TLS_RSA_WITH_AES_128_CBC_SHA is required in order to ensure compliance with RFC 5246.

These requirements will be revisited as new TLS versions are standardized by the IETF.

In a future version of this CPP TLS v1.2 will be required for all TOEs.

FCS_TLSS_EXT.1.2

The TSF shall deny connections from clients requesting SSL 2.0, SSL 3.0, TLS 1.0, and none.

Application Note 84

All SSL versions and TLS v1.0 shall be denied. Any TLS versions not selected in FCS_TLSS_EXT.1.1 should be selected here.

FCS_TLSS_EXT.1.3

The TSF shall perform RSA key establishment with key size 2048 bits, 3072 bits; generate EC Diffie-Hellman parameters over NIST curves secp256r1, secp384r1 and no other curves.

Application Note 88

If the ST lists a DHE or ECDHE ciphersuite in FCS_TLSS_EXT.1.1, the ST must include the Diffie-Hellman or NIST curves selection in the requirement. FMT_SMF.1 requires the configuration of the key agreement parameters in order to establish the security strength of the TLS connection.

6.1.3 Class FIA: Identification and Authentication

6.1.3.1 FIA_PMG_EXT.1 Password Management

FIA_PMG_EXT.1.1

The TSF shall provide the following password management capabilities for administrative passwords:

- a) Passwords shall be able to be composed of any combination of upper and lower case letters, numbers, and the following special characters: “!”, “@”, “#”, “\$”, “%”, “^”, “&”, “*”, “(”, “)”;
- b) Minimum password length shall be settable by the Security Administrator, and shall support passwords of 15 characters or greater.

Application Note 14

The ST author selects the special characters that are supported by TOE; they may optionally list additional special characters supported using the assignment. “Administrative passwords” refers to passwords used by administrators at the local console, over protocols that support passwords, such as SSH and HTTPS, or to grant configuration data that supports other SFRs in the Security Target.

6.1.3.2 FIA_UIA_EXT.1 User Identification and Authentication

FIA_UIA_EXT.1.1

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

- Display the warning banner in accordance with FTA_TAB.1;
- **Respond to ICMP Echo messages with an ICMP Echo Reply messages.**

FIA_UIA_EXT.1.2

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

Application Note 15

This requirement applies to users (administrators and external IT entities) of services available from the TOE directly, and not services available by connecting through the TOE. While it should be the case that few or no services are available to external entities prior to identification and authentication, if there are some available (perhaps ICMP echo) these should be listed in the assignment statement; otherwise “no other actions” should be selected.

Authentication can be password-based through the local console or through a protocol that supports passwords (such as SSH), or be certificate based (such as SSH, TLS).

For communications with external IT entities (e.g., an audit server or NTP server, for instance), such connections must be performed in accordance with FTP_ITC.1, whose protocols perform identification and authentication. This means that such communications (e.g., establishing the IPsec connection to the authentication server) would not have to be specified in the assignment, since establishing the connection “counts” as initiating the identification and authentication process.

6.1.3.3 FIA_UAU_EXT.2 Password-based Authentication Mechanism

FIA_UAU_EXT.2.1

The TSF shall provide a local password-based authentication mechanism, none to perform administrative user authentication.

Application Note 16

The assignment should be used to identify any additional local authentication mechanisms supported. Local authentication mechanisms are defined as those that occur through the local console; remote administrative sessions (and their associated authentication mechanisms) are specified in FTP_TRP.1.

6.1.3.4 FIA_UAU.7 Protected Authentication Feedback

FIA_UAU.7.1

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

Application Note 17

“Obscured feedback” implies the TSF does not produce a visible display of any authentication data entered by a user (such as the echoing of a password), although an obscured indication of progress may be provided (such as an asterisk for each character). It also implies that the TSF does not return any information during the authentication process to the user that may provide any indication of the authentication data.

6.1.3.5 FIA_X509_EXT.1 X.509 Certificate Validation

FIA_X509_EXT.1.1

The TSF shall validate certificates in accordance with the following rules:

- RFC 5280 certificate validation and certificate path validation.
- The certificate path must terminate with a trusted CA certificate.
- 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 a Certificate Revocation List (CRL) as specified in RFC 5280 Section 6.3.
- The TSF shall validate the extendedKeyUsage field according to the following rules:
 - 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) in the extendedKeyUsage field.
 - 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.
 - Client certificates presented for TLS shall have the Client Authentication purpose (id-kp 2 with OID 1.3.6.1.5.5.7.3.2) in the extendedKeyUsage field.

Application Note 18

FIA_X509_EXT.1.1 lists the rules for validating certificates. The ST author selects whether revocation status is verified using OCSP or CRLs. The trusted channel/path protocols require that certificates are used; this use requires that the extendedKeyUsage rules are verified.

The validation is expected to end in a trusted root CA certificate in a root store managed by the platform.

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.

Application Note 19

This requirement applies to certificates that are used and processed by the TSF and restricts the certificates that may be added as trusted CA certificates.

ST Application Note

- a) *FIA_X509_EXT.1 Assurance Activities were updated as specified by TD0093, TD0117, and TD0187.*

6.1.3.6 FIA_X509_EXT.2 X.509 Certificate Authentication

FIA_X509_EXT.2.1

The TSF shall use X.509v3 certificates as defined by RFC 5280 to support authentication for TLS, HTTPS, and no additional uses.

Application Note 20

The ST author's selection matches the selection of FTP_ITC.1.1. Certificates may optionally be used for trusted updates of system software (FPT_TUD_EXT.1) and for integrity verification (FPT_TST_EXT.2).

FIA_X509_EXT.2.2

When the TSF cannot establish a connection to determine the validity of a certificate, the TSF shall accept the certificate.

Application Note 21

Often a connection must be established to check the revocation status of a certificate -either to download a CRL or to perform a lookup using OCSP. The selection is used to describe the behavior in the event that such a connection cannot be established (for example, due to a network error). If the TOE has determined the certificate valid according to all other rules in FIA_X509_EXT.1, the behavior indicated in the selection determines the validity. The TOE must not accept the certificate if it fails any of the other validation rules in FIA_X509_EXT.1. If the administrator-configured option is selected by the ST Author, the ST Author also selects the corresponding function in FMT_SMF.1.

6.1.3.7 FIA_X509_EXT.3 X.509 Certificate Requests

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 and Common Name, Organization, Organizational Unit, Country.

Application Note 22

The public key is the public key portion of the public-private key pair generated by the TOE as specified in FCS_CKM.1(1).

FIA_X509_EXT.3.2

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

6.1.4 Class FMT: Security Management

6.1.4.1 FMT_MOF.1(1)/TrustedUpdate Management of Security Functions Behaviour

FMT_MOF.1.1(1)/TrustedUpdate

The TSF shall restrict the ability to enable the functions to perform manual update to Security Administrators.

Application Note 23

FMT_MOF.1(1)/TrustedUpdate restricts the initiation of manual updates to Security Administrators.

6.1.4.2 FMT_MOF.1(1)/Audit Management of Security Functions Behaviour

FMT_MOF.1.1(1)/Audit

The TSF shall restrict the ability to determine the behaviour of, modify the behaviour of the functions transmission of audit data to an external IT entity to Security Administrators.

Application Note 43

FMT_MOF.1(1)/Audit should always be chosen if the transmission protocol for transmission of audit data to an external IT entity as defined in FAU_STG_EXT.1.1 is configurable.

6.1.4.3 FMT_MOF.1(2)/Audit Management of Security Functions Behaviour

FMT_MOF.1.1(2)/Audit

The TSF shall restrict the ability to determine the behaviour of, modify the behaviour of the functions handling of audit data to Security Administrators.

Application Note 44

FMT_MOF.1(2)/Audit should only be chosen if the handling of audit data is configurable. The term 'handling of audit data' refers to the different options for selection and assignments in SFRs FAU_STG_EXT.1.2, FAU_STG_EXT.1.3 and FAU_STG_EXT.2.

6.1.4.4 FMT_MOF.1(1)/AdminAct Management of Security Behaviour

FMT_MOF.1.1(1)/AdminAct

The TSF shall restrict the ability to modify the behaviour of the functions TOE Security Functions to Security Administrators.

Application Note 45

FMT_MOF.1(1)/AdminAct should only be chosen if the behaviour of the TOE Security Functions is configurable.

6.1.4.5 FMT_MTD.1 Management of TSF Data

FMT_MTD.1.1

The TSF shall restrict the ability to manage the TSF Data to Security Administrators.

Application Note 24

The word "manage" includes but is not limited to create, initialize, view, change default, modify, delete, clear, and append. This SFR includes also the resetting of user passwords by the Security Administrator.

6.1.4.6 FMT_MTD.1/AdminAct Management of TSF Data

FMT_MTD.1.1/AdminAct

The TSF shall restrict the ability to modify, delete, generate/import the cryptographic keys to Security Administrators.

Application Note 48

FMT_MTD.1.1/AdminAct should only be chosen if cryptographic keys can be modified, deleted or generated/imported by the Security Administrator.

6.1.4.7 FMT_SMF.1 Specification of Management Functions

FMT_SMF.1.1

The TSF shall be capable of performing the following management functions:

- Ability to administer the TOE locally and remotely;
- Ability to configure the access banner;
- Ability to configure the session inactivity time before session termination or locking;
- Ability to update the TOE, and to verify the updates using digital signature capability prior to installing those updates;
- Ability to configure audit behavior;
- Ability to configure the cryptographic functionality.

Application Note 25

The TOE must provide functionality for both local and remote administration, including the ability to configure the access banner for FTA_TAB.1 and the session inactivity time(s) for FTA_SSL.3 & FTA_SSL.4. The item "Ability to update the TOE, and to verify the updates using digital signature capability prior to installing those updates" includes the relevant management functions from FMT_MOF.1(1)/TrustedUpdate, FMT_MOF.1(2)/TrustedUpdate (if included in the ST), FIA_X509_EXT.2.2 and FPT_TUD_EXT.2.2 (if included in the ST and if they include an administrator-configurable action). Similarly, the selection "Ability to configure audit behavior" includes the relevant management functions from FMT_MOF.1(1)/Audit, FMT_MOF.1(2)/Audit, FMT_MOF.1.1(1)/AdminAct, FMT_MOF.1.1(2)/AdminAct and FMT_MOF.1/LocSpace (for all of these SFRs that are included in the ST). If the TOE offers the ability for the administrator to configure the audit behaviour, configure the services available prior to identification or authentication, or if any of the cryptographic functionality on the TOE can be configured, then the ST author makes the appropriate choice or choices in the second selection, otherwise select "No other capabilities."

ST Application Note

FMT_SMF.1.1 was updated as specified by TD0090.

6.1.4.8 FMT_SMR.2 Restrictions on Security Roles

FMT_SMR.2.1

The TSF shall maintain the roles:

- Security Administrator.

FMT_SMR.2.2

The TSF shall be able to associate users with roles.

FMT_SMR.2.3

The TSF shall ensure that the conditions:

- The Security Administrator role shall be able to administer the TOE locally;
- The Security Administrator role shall be able to administer the TOE remotely

are satisfied.

Application Note 26

FMT_SMR.2.3 requires that a Security Administrator be able to administer the TOE through the local console and through a remote mechanism (IPsec, SSH, TLS, HTTPS).

6.1.5 Class FPT: Protection of the TSF

6.1.5.1 FPT_SKP_EXT.1 Protection of TSF Data (for reading of all symmetric keys)

FPT_SKP_EXT.1.1

The TSF shall prevent reading of all pre-shared keys, symmetric keys, and private keys.

Application Note 27

The intent of this requirement is for the device to protect keys, key material, and authentication credentials from unauthorized disclosure. This data should only be accessed for the purposes of their assigned security functionality, and there is no need for them to be displayed/accessed at any other time. This requirement does not prevent the device from providing indication that these exist, are in use, or are still valid. It does, however, restrict the reading of the values outright.

6.1.5.2 FPT_APW_EXT.1 Protection of Administrator Passwords

FPT_APW_EXT.1.1

The TSF shall store passwords in non-plaintext form.

FPT_APW_EXT.1.2

The TSF shall prevent the reading of plaintext passwords.

Application Note 28

The intent of the requirement is that raw password authentication data are not stored in the clear, and that no user or administrator is able to read the plaintext password through “normal” interfaces. An all-powerful administrator of course could directly read memory to capture a password but is trusted not to do so.

6.1.5.3 FPT_TST_EXT.1 TSF Testing

FPT_TST_EXT.1.1

The TSF shall run a suite of the following self-tests during initial start-up (on power on) to demonstrate the correct operation of the TSF: **BIOS checks, Cryptographic library functionality test, and firmware integrity checks.**

Application Note 29

It is expected that self-tests are carried out during initial start-up (on power on). Other options should only be used if the developer can justify why they are not carried out during initial start-up. It is expected that at least self-tests for verification of the integrity of the firmware and software as well as for the correct operation of cryptographic functions necessary to fulfil the SFRs will be performed. If not all self-test are performed during start-up multiple iterations of this SFR are used with the appropriate options selected. In future versions of this cPP the suite of self-tests will be required to contain at least mechanisms for measured boot including self-tests of the components which perform the measurement.

Application Note 30

If certificates are used by the self-test mechanism (e.g. for verification of signatures for integrity verification), certificates are validated in accordance with FIA_X509_EXT.1 and should be selected in FIA_X509_EXT.2.1. Additionally, FPT_TST_EXT.2 must be included in the ST.

6.1.5.4 FPT_TUD_EXT.1 Trusted Update

FPT_TUD_EXT.1.1

The TSF shall provide Security Administrators the ability to query the currently executing version of the TOE firmware/software and no other TOE firmware/software version.

Application Note 31

If a trusted update can be installed on the TOE with a delayed activation the version of both the currently executing image and the installed but inactive image must be provided. In this case the option 'the most recently installed version of the TOE firmware/software' needs to be chosen from the selection in FPT_TUD_EXT.1.1 and the TSS needs to describe how and when the inactive version becomes active. If all trusted updates become active as part of the installation process, only the currently executing version needs to be provided. In this case the option 'no other TOE firmware/software version' shall be chosen from the selection in FPT_TUD_EXT.1.1.

ST Application Note

FPT_TUD_EXT.1.1 was updated as specified by TD0154.

FPT_TUD_EXT.1.2

The TSF shall provide Security Administrators the ability to manually initiate updates to TOE firmware/software and no other update mechanism.

Application Note 32

The selection in FPT_TUD_EXT.1.2 distinguishes the support of automatic checking for updates and support of automatic updates. The first option refers to a TOE that checks whether a new update is available, communicates this to the administrator (e.g. through a message during an administrator session, through log files) but requires some action by the administrator to actually perform the update. The second option refers to a TOE that checks for updates and automatically installs them upon availability.

FPT_TUD_EXT.1.3

The TSF shall provide means to authenticate firmware/software updates to the TOE using a digital signature mechanism prior to installing those updates.

Application Note 33

The digital signature mechanism referenced in the selection of FPT_TUD_EXT.1.3 is one of the algorithms specified in FCS_COP.1(2). The published hash referenced in FPT_TUD_EXT.1.3 is generated by one of the functions specified in FCS_COP.1(3). The ST author should choose the mechanism implemented by the TOE; it is acceptable to implement both mechanisms.

Application Note 34

Future versions of this cPP will mandate the use of a digital signature mechanism for trusted updates.

Application Note 35

If certificates are used by the update verification mechanism, certificates are validated in accordance with FIA_X509_EXT.1 and should be selected in FIA_X509_EXT.2.1. Additionally, FPT_TUD_EXT.2 must be included in the ST.

Application Note 36

“Update” in the context of this SFR refers to the process of replacing a non-volatile, system resident software component with another. The former is referred to as the NV image, and the latter is the update image. While the update image is typically newer than the NV image, this is not a requirement. There are legitimate cases where the system owner may want to rollback a component to an older version (e.g. when the component manufacturer releases a faulty update, or when the system relies on an undocumented feature no longer present in the update). Likewise, the owner may want to update with the same version as the NV image to recover from faulty storage.

All discrete software components (e.g. applications, drivers, kernel, firmware) of the TSF, should be digitally signed by the corresponding manufacturer and subsequently verified by the mechanism performing the update. Since it is recognized that components may be signed by different manufacturers, it is essential that the update process verify that both the update and NV images were produced by the same manufacturer (e.g. by comparing public keys) or signed by legitimate signing keys (e.g. successful verification of certificates when using X.509 certificates).

6.1.5.5 FPT_STM.1 Reliable Time Stamps

FPT_STM.1.1

The TSF shall be able to provide reliable time stamps.

Application Note 37

The TSF does not provide reliable information about the current time at the TOE’s location by itself, but depends on external time and date information, either provided manually by the administrator or through the use of an NTP server. The term ‘reliable time stamps’ refers to the strict use of the time and date information, that is provided externally, and the logging of all changes to the time settings including information about the old and new time. With this information the real time for all audit data can be calculated.

6.1.6 Class FTA: TOE Access

6.1.6.1 FTA_SSL_EXT.1 TSF-initiated Session Locking

FTA_SSL_EXT.1.1

The TSF shall, for local interactive sessions,

- terminate the session

after a Security Administrator-specified time period of inactivity.

6.1.6.2 FTA_SSL.3 TSF-initiated Termination

FTA_SSL.3.1

The TSF shall terminate a remote interactive session after a Security Administrator-configurable time interval of session inactivity.

6.1.6.3 FTA_SSL.4 User-initiated Termination

FTA_SSL.4.1

The TSF shall allow Administrator-initiated termination of the Administrator's own interactive session.

6.1.6.4 FTA_TAB.1 Default TOE Access Banners

FTA_TAB.1.1

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

Application Note 38

This requirement is intended to apply to interactive sessions between a human user and a TOE. IT entities establishing connections or programmatic connections (e.g., remote procedure calls over a network) are not required to be covered by this requirement.

6.1.7 Class FTP: Trusted Path/Channels

6.1.7.1 FTP_ITC.1 Inter-TSF Trusted Channel

FTP_ITC.1.1

The TSF shall be capable of using TLS, HTTPS to provide a trusted communication channel between itself and authorized IT entities supporting the following capabilities: audit server, **Pulse One management server** that is logically distinct from other communication channels and provides assured identification of its end points and protection of the channel data from disclosure and detection of modification of the channel data.

FTP_ITC.1.2

The TSF shall permit the TSF, or 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 **audit server communications, Pulse One management server communication.**

Application Note 39

The intent of the above requirement is to provide a means by which a cryptographic protocol may be used to protect external communications with authorized IT entities that the TOE interacts with to perform its functions. The TOE uses at least one of the listed protocols for communications with the server that collects the audit information. If it communicates with an authentication server (e.g., RADIUS), then the ST author chooses "authentication server" in FTP_ITC.1.1 and this connection must be capable of being protected by one of the listed protocols. If other authorized IT entities (e.g., NTP server) are protected, the ST author makes the appropriate assignments (for those entities) and selections (for the protocols that are used to protect those connections). The ST author selects the mechanism or mechanisms supported by the TOE, and then ensures that the detailed protocol requirements in Appendix B corresponding to their selection are included in the ST.

While there are no requirements on the party initiating the communication, the ST author lists in the assignment for FTP_ITC.1.3 the services for which the TOE can initiate the communication with the authorized IT entity.

The requirement implies that not only are communications protected when they are initially established, but also on resumption after an outage. It may be the case that some part of the TOE setup involves manually setting up tunnels to protect other communication, and if after an outage the TOE attempts to re-establish the communication automatically with (the necessary) manual intervention, there may be a window created where an attacker might be able to gain critical information or compromise a connection.

ST Application Note

Application Note 39 was updated as specified by TD0126.

6.1.7.2 FTP_TRP.1 Trusted Path

FTP_TRP.1.1

The TSF shall be capable of using TLS, HTTPS to provide a communication path between itself and authorized remote administrators that is logically distinct from other communication paths and provides assured identification of its end points and protection of the communicated data from disclosure and provides detection of modification of the channel data.

FTP_TRP.1.2

The TSF shall permit remote administrators to initiate communication via the trusted path.

FTP_TRP.1.3

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

Application Note 40

This requirement ensures that authorized remote administrators initiate all communication with the TOE via a trusted path, and that all communication with the TOE by remote administrators is performed over this path. The data passed in this trusted communication channel are encrypted as defined by the protocol chosen in the first selection. The ST author selects the mechanism or mechanisms supported by the TOE,

and then ensures that the detailed protocol requirements in Appendix B corresponding to their selection are included in the ST.

6.2 Security Assurance Requirements

This Security Target is conformant with the assurance requirements specified in the cPP.

Table 9: Assurance Requirements	
Assurance Class	Assurance Component
Security Target (ASE)	Conformance claims (ASE_CCL.1)
	Extended components definition (ASE_ECD.1)
	ST introduction (ASE_INT.1)
	Security objectives for the operational environment (ASE_OBJ.1)
	Stated security requirements (ASE_REQ.1)
	Security Problem Definition (ASE_SPD.1)
	TOE summary specification (ASE_TSS.1)
Development (ADV)	Basic functional specification (ADV_FSP.1)
Guidance documents (AGD)	Operational user guidance (AGD_OPE.1)
	Preparative procedures (AGD_PRE.1)
Life cycle support (ALC)	Labeling of the TOE (ALC_CMC.1)
	TOE CM coverage (ALC_CMS.1)
Tests (ATE)	Independent testing –sample (ATE_IND.1)
Vulnerability assessment (AVA)	Vulnerability survey (AVA_VAN.1)

6.2.1 Extended Security Assurance Requirements

These requirements are taken directly from the cPP.

6.2.1.1 ASE: Security Target

The ST is evaluated as per ASE activities defined in the CEM. In addition, there may be Evaluation Activities specified within the [SD] that call for necessary descriptions to be included in the TSS that are specific to the TOE technology type.

[NDcPP, Appendix D] provides a description of the information expected to be provided regarding the quality of entropy in the random bit generator.

The requirements for exact conformance of the Security Target are described in [NDcPP, 2] and in [SD, 3.1].

6.2.1.1.1 Conformance Claims (ASE_CCL.1)

The table below indicates the actions to be taken for particular ASE_CCL.1 elements in order to determine exact conformance with a cPP.

Table 10: Conformance Claims	
ASE_CCL.1 Element	Evaluator Action
ASE_CCL.1.8C	The evaluator shall check that the statements of security problem definition in the PP and ST are identical.
ASE_CCL.1.9C	The evaluator shall check that the statements of security objectives in the PP and ST are identical.

<p>ASE_CCL.1.10C</p>	<p>The evaluator shall check that the statements of security requirements in the ST include all the mandatory SFRs in the cPP, and all of the selection-based SFRs that are entailed by selections made in other SFRs (including any SFR iterations added in the ST). The evaluator shall check that if any other SFRs are present in the ST (apart from iterations of SFRs in the cPP) then these are taken only from the list of optional SFRs specified in the cPP (the cPP will not necessarily include optional SFRs, but may do so). If optional SFRs from the cPP are included in the ST then the evaluator shall check that any selection-based SFRs entailed by the optional SFRs adopted are also included in the ST.</p>
----------------------	---

6.2.1.1.2 TOE Summary Specification (ASE_TSS.1)

Evaluation Activities

The TOE summary specification shall describe how the TOE meets each SFR. In the case of entropy analysis the TSS is used in conjunction with required supplementary information on Entropy.

6.2.1.2 ADV: Development

The design information about the TOE is contained in the guidance documentation available to the end user as well as the TSS portion of the ST, and any required supplementary information required by this cPP that is not to be made public.

6.2.1.2.1 Basic Functional Specification (ADV_FSP.1)

The functional specification describes the TOE Security Functions Interfaces (TSFIs). It is not necessary to have a formal or complete specification of these interfaces. Additionally, because TOEs conforming to this cPP will necessarily have interfaces to the Operational Environment that are not directly invocable by TOE users, there is little point specifying that such interfaces be described in and of themselves since only indirect testing of such interfaces may be possible. For this cPP, the Evaluation Activities for this family focus on understanding the interfaces presented in the TSS in response to the functional requirements and the interfaces presented in the AGD documentation. No additional “functional specification” documentation is necessary to satisfy the Evaluation Activities specified in the SD.

The Evaluation Activities in the SD are associated with the applicable SFRs; since these are directly associated with the SFRs, the tracing in element ADV_FSP.1.2D is implicitly already done and no additional documentation is necessary.

Evaluation Activities

The evaluator shall check the interface documentation to ensure it describes the purpose and method of use for each TSFI that is identified as being security relevant.

In this context, TSFI are deemed security relevant if they are used by the administrator to configure the TOE, or to perform other administrative functions (e.g., audit review or performing updates). Additionally, those interfaces that are identified in the ST, or guidance documentation, as adhering to the security policies (as presented in the SFRs), are also considered security relevant. The intent, is that these interfaces will be adequately tested, and having an understanding of how these interfaces are used in the TOE is necessary to ensure proper test coverage is applied.

The evaluator shall check the interface documentation to ensure it identifies and describes the parameters for each TSFI that is identified as being security relevant.

The documents to be examined for this assurance component in an evaluation are therefore the Security Target, AGD documentation, and any supplementary information required by the cPP for aspects such as entropy analysis or cryptographic key management architecture¹: no additional “functional specification” documentation is necessary to satisfy the Evaluation Activities. The interfaces that need to be evaluated are also identified by reference to the assurance activities listed for each SFR, and are expected to be identified in the context of the Security Target, AGD documentation, and any supplementary information required by the cPP rather than as a separate list specifically for the purposes of CC evaluation. The direct identification of documentation requirements and their assessment as part of the Evaluation Activities for each SFR also means that the tracing required in ADV_FSP.1.2D is treated as implicit, and no separate mapping information is required for this element.

However, if the evaluator is unable to perform some other required Evaluation Activity because there is insufficient design and interface information, then the evaluator is entitled to conclude that an adequate functional specification has not been provided, and hence that the verdict for the ADV_FSP.1 assurance component is a ‘fail’.

6.2.1.3 AGD: Guidance Documentation

The guidance documents will be provided with the ST. Guidance must include a description of how the IT personnel verifies that the Operational Environment can fulfill its role for the security functionality. The documentation should be in an informal style and readable by the IT personnel.

Guidance must be provided for every operational environment that the product supports as claimed in the ST. This guidance includes:

- instructions to successfully install the TSF in that environment; and
- instructions to manage the security of the TSF as a product and as a component of the larger operational environment; and
- instructions to provide a protected administrative capability.

Guidance pertaining to particular security functionality must also be provided; requirements on such guidance are contained in the Evaluation Activities specified in the SD.

6.2.1.3.1 Operational User Guidance (AGD_OPE.1)

The operational user guidance does not have to be contained in a single document. Guidance to users, administrators and application developers can be spread among documents or web pages.

The developer should review the Evaluation Activities contained in the SD to ascertain the specifics of the guidance that the evaluator will be checking for. This will provide the necessary information for the preparation of acceptable guidance.

Evaluation Activities

The evaluator shall check the requirements below are met by the guidance documentation.

Guidance documentation shall be distributed to administrators and users (as appropriate) as part of the TOE, so that there is a reasonable guarantee that administrators and users are aware of the existence and role of the documentation in establishing and maintaining the evaluated configuration.

¹ The Security Target and AGD documentation are public documents. Supplementary information may be public or proprietary: the cPP and/or Evaluation Activity descriptions will identify where such supplementary documentation is permitted to be proprietary and non-public.

Guidance documentation must be provided for every Operational Environment that the product supports as claimed in the Security Target and must adequately address all platforms claimed for the TOE in the Security Target.

The contents of the guidance documentation will be verified by the Evaluation Activities defined below and as appropriate for each individual SFR in section 2 above.

In addition to SFR-related Evaluation Activities, the following information is also required.

- a) The guidance documentation shall contain instructions for configuring any cryptographic engine associated with the evaluated configuration of the TOE. It shall provide a warning to the administrator that use of other cryptographic engines was not evaluated nor tested during the CC evaluation of the TOE.
- b) The documentation must describe the process for verifying updates to the TOE by verifying a digital signature. The evaluator shall verify that this process includes the following steps:
 - 1) Instructions for obtaining the update itself. This should include instructions for making the update accessible to the TOE (e.g., placement in a specific directory).
 - 2) Instructions for initiating the update process, as well as discerning whether the process was successful or unsuccessful. This includes generation of the hash/digital signature.
- c) The TOE will likely contain security functionality that does not fall in the scope of evaluation under this cPP. The guidance documentation shall make it clear to an administrator which security functionality is covered by the Evaluation Activities.

6.2.1.3.2 Preparative Procedures (AGD_PRE.1)

As with the operational guidance, the developer should look to the Evaluation Activities to determine the required content with respect to preparative procedures.

Evaluation Activities

The evaluator shall check the requirements below are met by the preparative procedures.

The contents of the preparative procedures will be verified by the Evaluation Activities defined below and as appropriate for each individual SFR in section 2 above.

Preparative procedures shall be distributed to administrators and users (as appropriate) as part of the TOE, so that there is a reasonable guarantee that administrators and users are aware of the existence and role of the documentation in establishing and maintaining the evaluated configuration.

The contents of the preparative procedures will be verified by the Evaluation Activities defined below and as appropriate for each individual SFR in section 2 above.

In addition to SFR-related Evaluation Activities, the following information is also required.

Preparative procedures must include a description of how the administrator verifies that the operational environment can fulfil its role to support the security functionality (including the requirements of the Security Objectives for the Operational Environment specified in the Security Target). The documentation should be in an informal style and should be written with sufficient detail and explanation that they can be understood and used by the target audience (which will typically include IT staff who have general IT experience but not necessarily experience with the TOE product itself).

Preparative procedures must be provided for every Operational Environment that the product supports as claimed in the Security Target and must adequately address all platforms claimed for the TOE in the Security Target.

The preparative procedures must include

- a) instructions to successfully install the TSF in each Operational Environment; and
- b) instructions to manage the security of the TSF as a product and as a component of the larger operational environment; and
- c) instructions to provide a protected administrative capability.

6.2.1.4 Class ALC: Life-cycle Support

At the assurance level provided for TOEs conformant to this cPP, life-cycle support is limited to end-user-visible aspects of the life-cycle, rather than an examination of the TOE vendor's development and configuration management process. This is not meant to diminish the critical role that a developer's practices play in contributing to the overall trustworthiness of a product; rather, it is a reflection on the information to be made available for evaluation at this assurance level.

6.2.1.4.1 Labelling of the TOE (ALC_CMC.1)

This component is targeted at identifying the TOE such that it can be distinguished from other products or versions from the same vendor and can be easily specified when being procured by an end user. A label could consist of a "hard label" (e.g., stamped into the metal, paper label) or a "soft label" (e.g., electronically presented when queried).

The evaluator performs the CEM work units associated with ALC_CMC.1.

6.2.1.4.2 TOE CM Coverage (ALC_CMS.1)

Given the scope of the TOE and its associated evaluation evidence requirements, the evaluator performs the CEM work units associated with ALC_CMS.1.

6.2.1.5 Class ATE: Tests

Testing is specified for functional aspects of the system as well as aspects that take advantage of design or implementation weaknesses. The former is done through the ATE_IND family, while the latter is through the AVA_VAN family. For this cPP, testing is based on advertised functionality and interfaces with dependency on the availability of design information. One of the primary outputs of the evaluation process is the test report as specified in the following requirements.

6.2.1.5.1 Independent Testing – Conformance (ATE_IND.1)

Testing is performed to confirm the functionality described in the TSS as well as the guidance documentation (includes "evaluated configuration" instructions). The focus of the testing is to confirm that the requirements specified in Section 5 are being met. The Evaluation Activities in the SD identify the specific testing activities necessary to verify compliance with the SFRs. The evaluator produces a test report documenting the plan for and results of testing, as well as coverage arguments focused on the platform/TOE combinations that are claiming conformance to this cPP.

Evaluation Activities

The evaluator shall examine the TOE to determine that the test configuration is consistent with the configuration under evaluation as specified in the ST.

The evaluator shall examine the TOE to determine that it has been installed properly and is in a known state.

The evaluator shall prepare a test plan that covers all of the testing actions for ATE_IND.1 in the CEM and in the SFR-related Evaluation Activities. While it is not necessary to have one test case per test listed in an Evaluation Activity, the evaluator must show in the test plan that each applicable testing requirement in the SFR-related Evaluation Activities is covered.

The test plan identifies the platforms to be tested, and for any platforms not included in the test plan but included in the ST, the test plan provides a justification for not testing the platforms. This justification must address the differences between the tested platforms and the untested platforms, and make an argument that the differences do not affect the testing to be performed. It is not sufficient to merely assert that the differences have no affect; rationale must be provided. If all platforms claimed in the ST are tested, then no rationale is necessary.

The test plan describes the composition and configuration of each platform to be tested, and any setup actions that are necessary beyond what is contained in the AGD documentation. It should be noted that the evaluator is expected to follow the AGD documentation for installation and setup of each platform either as part of a test or as a standard pre-test condition. This may include special test drivers or tools. For each driver or tool, an argument (not just an assertion) should be provided that the driver or tool will not adversely affect the performance of the functionality by the TOE and its platform. This also includes the configuration of any cryptographic engine to be used (e.g. for cryptographic protocols being evaluated).

The test plan identifies high-level test objectives as well as the test procedures to be followed to achieve those objectives, and the expected results.

The test report (which could just be an updated version of the test plan) details the activities that took place when the test procedures were executed, and includes the actual results of the tests. This shall be a cumulative account, so if there was a test run that resulted in a failure, so that a fix was then installed and then a successful re-run of the test was carried out, then the report would show a “fail” result followed by a “pass” result (and the supporting details), and not just the “pass” result².

6.2.1.6 Class AVA: Vulnerability Assessment

For the first generation of this cPP, the iTC is expected to survey open sources to discover what vulnerabilities have been discovered in these types of products and provide that content into the AVA_VAN discussion. In most cases, these vulnerabilities will require sophistication beyond that of a basic attacker. This information will be used in the development of future protection profiles.

6.2.1.6.1 Vulnerability Survey (AVA_VAN.1)

Appendix A in [SD] provides a guide to the evaluator in performing a vulnerability analysis.

Evaluation Activities

The evaluator shall document their analysis and testing of potential vulnerabilities with respect to this requirement. This report could be included as part of the test report for ATE_IND, or could be a separate document.

The evaluator formulates hypotheses in accordance with process defined in Appendix A. The evaluator documents the flaw hypotheses generated for the TOE in the report in accordance with the guidelines in

² It is not necessary to capture failures that were due to errors on the part of the tester or test environment. The intention here is to make absolutely clear when a planned test resulted in a change being required to the originally specified test configuration in the test plan, to the evaluated configuration identified in the ST and guidance documentation, or to the TOE itself.

Appendix A.5. The evaluator shall then perform vulnerability analysis in accordance with Appendix A.4. The results of the analysis shall be documented in the report according to Appendix A.5.

7. TOE Summary Specification

This section provides evaluators and potential consumers of the TOE with a high-level description of each SFR, thereby enabling them to gain a general understanding of how the TOE is implemented. These descriptions are intentionally not overly detailed, thereby disclosing no proprietary information. These sections refer to SFRs defined in Section 6, Security Requirements.

The TOE consists of the following Security Functions:

- Security Audit
- Cryptographic Support
- Identification and Authentication
- Security Management
- Protection of the TSF
- TOE Access
- Trusted Path/Channels

7.1 Security Audit

7.1.1 Audit Generation

The TSF generates audit records for the following events:

- Startup and shutdown of the audit function
- Administrative login and logout events
- Security related configuration changes
- Generation of a CSR and associated keypair
- Installation of a certificate
- Resetting passwords
- Low audit storage space available
- Failure to establish a HTTPS/TLS session
- Failure to establish a TLS session
- All use of the identification and authentication mechanism (local and remote connections to the TSF)
- Unsuccessful attempts to validate a certificate
- Initiation of a software update
- Result of a software update
- Changes to the time
- Modification of the behavior of the TSF
- Failure of self-tests
- Initiation and termination of the trusted channel
- Initiation and termination of the trusted path

Each audit record includes the date and time, type, subject identity (IP address, hostname, and/or username), the outcome (success or failure), and any additional information specified in column three of Table 8.

FAU_GEN.1, FAU_GEN.2

7.1.2 Audit Storage

By default the TSF allocates 200 MB to local audit storage; however, the administrator can configure the amount of space allocated to local audit storage, up to 500 MB. The TSF divides the local audit storage

between two audit files. When the current audit file reaches capacity; the TSF deletes the inactive log file, creates a new log file, switches logging to the new log file, and generates an audit log indicating that a log file reached capacity. The administrator can also configure the TSF to generate an audit log warning when the current audit file reaches a specified percentage of its capacity.

The TSF protects audit data from unauthorized modification and deletion through the restrictive administrative interfaces. The filesystem of the TSF is not exposed to the administrative user over the HTTPs GUI or the local CLI. The administrative user must be positively identified and authenticated prior to being allowed to clear the local audit log or change audit settings.

The TSF implements the Syslog protocol and Syslog over TLS according to RFCs 5424 and 5425 respectively. The trusted channel with the Syslog server is described in greater detail in Section 7.2.4.

FAU_STG.1, FAU_STG_EXT.1, FAU_STG_EXT.3

7.2 Cryptographic Support

7.2.1 Cryptographic Key Generation

The TSF supports the generation of RSA 2048 bit and 3072 bit keys for TLS client authentication, TLS server authentication, and RSA key encapsulation.

The TSF generates ECDSA P-256 and P-384 keys for TLS client authentication, TLS server authentication, and TLS ECDHE key establishment.

The TSF generates DSA 2048 bit modp keys for TLS DHE key establishment.

FCS_CKM.1

The TSF utilizes elliptic curve key agreement when an ECDHE TLS ciphersuite is negotiated. The TSF utilizes RSA based key encapsulation when any other TLS ciphersuite is negotiated.

FCS_CKM.2

The TSF stores the following persistent keys on internal Hard Disk Drives in plaintext:

- HTTPs/TLS Private Host Key – generated using the DRBG and FCS_CKM.1 or entered by the Security Administrator.
- Syslog/TLS Private Client Key – generated using the DRBG and FCS_CKM.1 or entered by the Security Administrator.
- HAWK secret key – Provided by the Pulse One server during registration and during the automated new Credentials process.

The TSF stores loads the persistent keys into RAM when they are used and the TSF also stores the following ephemeral keys in RAM:

- TLS Session keys – Established according to FCS_CKM.2 and derived using the TLS KDF
- DRBG State – Derived from the entropy source

The HTTPs/TLS Private Host Key and the Syslog/TLS Private Client key are zeroized from the disk when the Security Administrator deletes the key, replaces the key, or zeroizes the entire TOE. These keys are zeroized from RAM when the HTTP or Syslog process terminates.

The TLS Session keys are zeroized from RAM when the associated TLS session is terminated.

The DRBG state is zeroized when the TSF is shutdown or restarted.

The TSF zeroizes keys in RAM by writing zeros to the memory location three times and performing a read verify to ensure that the memory location was set to all zeros. If the read verify fails, the TSF repeats the zeroization process.

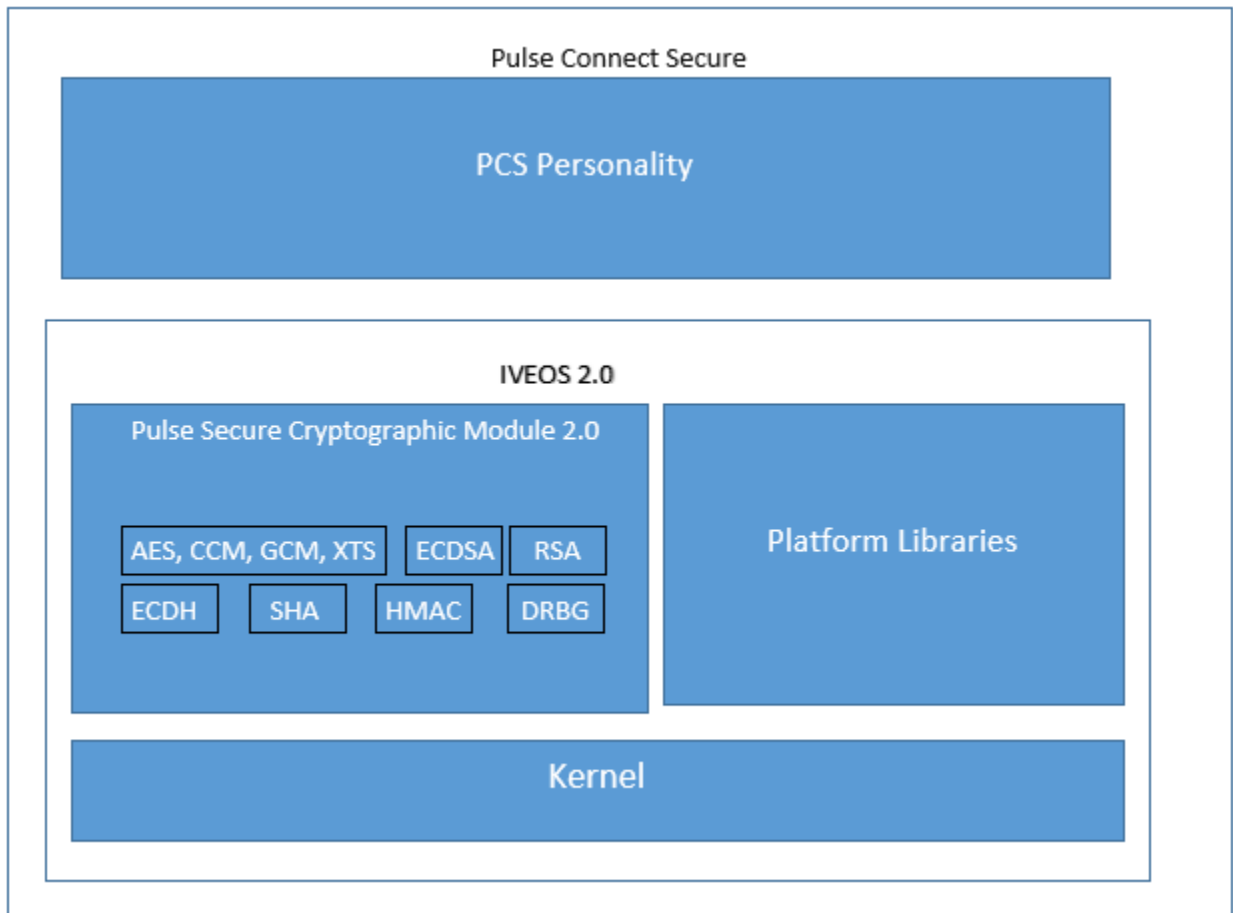
The TSF zeroizes keys on the hard disk drives by overwriting the file location with data from /dev/urandom three times. Each overwrite calls /dev/urandom ensuring that a different pseudo random pattern is used each time.

FCS_CKM.4

7.2.2 Cryptographic Operations

The TSF's use of cryptographic is architected as show below.

Figure1



The TSF utilizes the following CAVP certified cryptographic algorithms:

Table 11: Cryptographic Algorithms		
SFR	Description	Cert
FCS_CKM.1	RSA 2048/3072-bit Key Generation	2345
	ECDSA P-256/P-384 Key Generation	1026
FCS_CKM.2	RSA Key Establishment	Vender Affirmed
	Elliptic Curve Diffie-Hellman SP800-56A Key Agreement	1270

Table 11: Cryptographic Algorithms		
SFR	Description	Cert
FCS_COP.1(1)	AES 128/256-bit CBC/GCM Encryption/Decryption	4334
FCS_COP.1(2)	RSA 2048/3072-bit Signature Generation, Signature Verification with SHA-1/256/384/512	2345
	ECDSA P-256/P-384 Signature Generation, Signature Verification with SHA-1/256/384/512	1026
FCS_COP.1(3)	SHA-1/256/384/512	3577
FCS_COP.1(4)	HMAC-SHA-1/256/384	2880
FCS_RBG_EXT.1	CTR_DRBG using AES 256	1384

The TSF seed the CTR_DRBG using 512-bits of data that contains at least 256 bits of entropy. The TSF gathers and pools entropy from two hardware noise sources: interrupt timings and hard disk timings.

The TSF utilizes the hash algorithms as describe below in Table 12.

Table 12: Hash Usage	
Hash	Use
SHA-1	HMAC as described in Table 13, Hashing for Digital Signatures
SHA-256	HMAC as described in Table 13, Hashing for Digital Signatures, File integrity checking, Password Obfuscation
SHA-384	HMAC as described in Table 13, Hashing for Digital Signatures
SHA-512	Hashing for Digital Signatures

The TSF utilizes the HMAC algorithm as describe below in Table 13.

Table 13: HMAC Usage				
Hash	Use	Key Size	Block Size	MAC length
SHA-1	TLSv1.1 Master Secret Derivation	128 bits – ECDHE P-256	512 bits	160 bits
		192 bits – RSA & ECDHE P-384		
		1024 bits – DHE 2048		
SHA-256	TLSv1.2 Master Secret Derivation	128 bits – ECDHE P-256	512 bits	256 bits
		192 bits – RSA & ECDHE P-384		
		1024 bits – DHE		
SHA-384	TLSv1.2 Master Secret Derivation	256 bits – ECDHE P-256	1024 bits	384 bits
		384 bits – RSA & ECDHE P-384		
		2048 bits – DHE 2048		
SHA-1	TLSv1.1 Key Block Derivation	192 bits	512 bits	160 bits
SHA-256	TLSv1.2 Key Block Derivation	384 bits	512 bits	256 bits
SHA-384	TLSv1.2 Key Block Derivation	384 bits	1024 bits	384 bits
SHA-1	TLS Message Authentication	160 bits	512 bits	160 bits
SHA-256	TLS Message Authentication	256 bits	512 bits	256 bits
SHA-384	TLS Message Authentication	384 bits	1024 bits	384 bits

Table 13: HMAC Usage				
Hash	Use	Key Size	Block Size	MAC length
SHA-256	HAWK HMAC Generation	352 bits	512 bits	256 bits

FCS_COP.1(1), FCS_COP.1(2), FCS_COP.1(3), FCS_COP.1(4), FCS_RBG_EXT.1

7.2.3 HTTPs Protocol

The TSF implements the server and client sides of the HTTPs protocol according to RFC 2818 by using a TLS session in place of a TCP connection. The TLS server protocol is described in Section 7.2.5 and the TLS client protocol is described in Section 7.2.4. In either instance, the TSF will not establish the connection if the peer certificate does not successfully authenticate the peer according to X.509 authentication described in Section 7.3.5.

When acting as a server, the TSF listens on port 443 for HTTPs connections. The TSF uses HTML over HTTPs to present the administrative users with a secure management interface described in Section 7.4. The TSF uses TLS to provide a secure connection between the TSF and remote Security Administrators.

When acting as a client, the TSF uses HTTPs to establish a trusted channel with the Pulse One management server.

FCS_HTTPS_EXT.1

7.2.4 TLS Client Protocol

The TSF implements a TLSv1.1 and TLSv1.2 client to secure communications with the Syslog server and the Pulse One server. The TSF supports and proposes the following ciphersuites and extensions in the ClientHello Message:

- Ciphersuites:
 - TLS_RSA_WITH_AES_128_CBC_SHA
 - TLS_RSA_WITH_AES_256_CBC_SHA
 - TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
 - TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
 - TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
 - TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
 - TLS_RSA_WITH_AES_128_CBC_SHA256
 - TLS_RSA_WITH_AES_256_CBC_SHA256
 - TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
 - TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
 - TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
 - TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
 - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
 - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- Ciphersuites for communication with Pulse One (FCS_TLSC_EXT.1):
 - TLS_RSA_WITH_AES_128_CBC_SHA
 - TLS_RSA_WITH_AES_256_CBC_SHA256
 - TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
 - TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
 - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384

- Signature Algorithms:
 - RSA with SHA-1/256/384/512
 - ECDSA SHA-1/256/384
- Supported Elliptic Curves:
 - secp256r1
 - secp384r1
- Supported Point Formats
 - Uncompressed

The Security Administrator can configure the TSF so it will only negotiate TLSv1.2. The Security Administrator can enable and disable individual ciphersuites as well as specifying the preferred ordering of ciphersuites. The TSF also allows the Security Administrator to enable or disable the supported elliptic curves.

When the Syslog server sends the Certificate Request message, the TSF replies with a Client Certificate message. The Client Certificate message includes the certificate that the Security Administrator configured to authenticate to the Syslog server.

When the TSF connects to the Pulse One management server, the TSF does not utilize TLS Client Certificate authentication. The TSF uses HAWK authentication described in Section 7.7.1 to authenticate the connection to the Pulse One server.

The TSF does not support certificate pinning and determines if the certificate is valid for the specified server based on the DNS name or IP address of the server as described in Section 7.3.5.

FCS_TLSC_EXT.1, FCS_TLSC_EXT.2

7.2.5 TLS Server Protocol

The TSF supports TLSv1.1 and TLSv1.2 for HTTPs/TLS. If the TSF receives a ClientHello message that requests TLSv1.0 or earlier, the TSF sends a fatal handshake_failure message and terminates the connection. When the TSF is configured with a server certificate with an RSA key, the TSF supports following TLS ciphersuites are supported:

- TLS_RSA_WITH_AES_128_CBC_SHA
- TLS_RSA_WITH_AES_256_CBC_SHA
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
- TLS_RSA_WITH_AES_128_CBC_SHA256
- TLS_RSA_WITH_AES_256_CBC_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384

When the TSF is configured with a server certificate with an ECDSA key, the TSF supports following TLS ciphersuites are supported:

- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384

When the TSF selects a DHE ciphersuite, it sends the client Diffie-Hellman 2048-bit modp key agreement parameters.

When the TSF selects an ECDHE ciphersuite, it sends the client secp256r1 or secp384r1 key agreement parameters. The TSF prefers secp256r1 if the client indicates support for both curves in the ClientHello message.

The Security Administrator can configure the TSF so it only accepts TLSv1.2 connections. The Security Administrator can enable and disable individual ciphersuites as well as specifying the preferred ordering of ciphersuites.

FCS_TLSS_EXT.1

7.3 Identification and Authentication

7.3.1 Password Management

The TSF supports administrator password composition to include any combination of upper and lower case letters, numbers, and the following special characters “!”, “@”, “#”, “\$”, “%”, “^”, “&”, “*”, “(”, “)”, and the complete set of standard printable ASCII characters (values 0x20 – 0x7E) with a minimum length settable by the administrator and support 15 characters or greater.

FIA_PMG_EXT.1.1

7.3.2 User Identification and Authentication

The TSF utilizes HTTPs as described in Section 7.2.3 to a secure remote administration web UI. When connecting over HTTPs, the TSF presents Security Administrators with a username and password prompt; however the Security Administrator can choose to authenticate using an X.509 certificate. In this case, the TSF forces a TLS Renegotiation. The TSF includes the Certificate Request message as part of the handshake so the client will send a Client Certificate message to authenticate the Security Administrator. The Security Administrator is considered authenticated if the username and password match, or if the username, certificate, and signature match.

The TSF utilizes a local serial CLI which presents Security Administrators with a username and password prompt. The Security Administrator is considered authenticated if the username and password provided match the credentials configured in the TSF.

Prior to successful identification and authentication, the TSF displays the TOE access banner specified in FTA_TAB.1 and responds to ICMP Echo messages with ICMP Echo Reply messages.

FIA_UIA_EXT.1, FIA_UAU_EXT.2

7.3.3 Protected Authentication Feedback

When the user is entering their password over the local console, the TSF does not echo any characters back.

FIA_UAU.7.1

7.3.4 X.509 Certificate Validation

When a certificate is used (to identify the TSF or identify an external entity to the TSF), the TSF verifies certificates by checking the following:

1. The current date between the “Valid from” and “Valid to” dates
2. The certificate is not listed on the CRL. If the TSF has a cached response that has not expired, the TSF uses the cached response in lieu of querying the CRL serverCL

3. The certificate chain is valid:
 - Each certificate in the certificate chain passes the checks described in #1 and #2.
 - Each certificate (other than the first certificate) in the certificate chain has the Subject Type=CA flag set.
 - Each certificate is signed by:
 - a certificate in the certificate chain, or
 - a trusted root CA that has been installed in the TSF

The TSF verifies the validity of a certificate when:

- A HTTPs client establishes a TLS connection (HTTPs Server Certificate)
- A HTTPs client presents a client authentication certificate
- The TSF verifies the server certificate of the Syslog server
- The TSF uses its client certificate to authenticate to the Syslog server

If the Security Administrator loads a certificate with a Subject Type=CA, the TSF does not validate the certificate path.

FIA_X509_EXT.1

7.3.5 X.509 Certificate Authentication

When establishing a connection to the Syslog server, the TSF uses the certificate presented by the Syslog server to verify the server's identity.

The TSF establishes reference identifiers for the remote server as follows:

- When the server is specified using a domain name, the TSF verifies that the domain name matches a Subject Alternative Name DNS Name field in the certificate using exact or wildcard matching specified in Section 3.1 of RFC 2818. If the certificate does not contain any Subject Alternative Name fields, the TSF matches the domain name against the Common Name in the certificate.
- When the server is specified using an IP address, the TSF verifies that the IP address exactly matches a Subject Alternative Name IP Address field in the certificate using the rules specified in Section 3.1 of RFC 2818. If the certificate does not contain any Subject Alternative Name fields, the TSF matches the IP address against the Common Name in the certificate.

Once the TSF has verified that the certificate identifiers are valid for the Syslog server, the TSF verifies the validity of the certificate as described in Section 7.3.4.

The TSF presents its own certificate to the Syslog server. This certificate is configured specifically for authentication to the Syslog server by the Security Administrator.

When a user connects over HTTPs, the TSF presents the certificate that the Security Administrator configured for use with HTTPs. If the user proceeds with certificate authentication, the TSF verifies that the certificate presented by the user is the same certificate that is configured for the provided username.

If the TSF cannot contact the CRL server or the server does not respond, the TSF logs the failure and considers the certificate valid. If any of the other Section 7.3.4 validity checks fail, the TSF rejects the certificate and does not establish the connection.

FIA_X509_EXT.2

7.3.6 X.509 Certificate Requests

The TSF allows Security Administrators to generate Certificate Signing Requests. The TSF requires the Security Administrator to specify the following values:

- Common Name
- Organization
- Locality
- State
- Country
- Key Type (RSA or ECDSA)
- Key Length (2048, 3072, P-256, or P-384)

The TSF allows the Security Administrator to specify an Organization Unit and additional random data used when generating the key pair.

FIA_X509_EXT.3

7.4 Security Management

The TSF implements the Security Administrator role to authorized administrators of the TOE. The TSF allows the Security Administrators to administer the TSF via a local CLI, and a web UI. The TSF accepts configuration updates from a Pulse One management server, which is a trusted IT entity. The TSF permission restrict access to these management functions to users that have been identified, authenticated, and authorized with the Security Administrator role. The web UI, the local console, and Pulse One management server allow the Security Administrator to perform the following TSF management functions:

- Verify/Install Firmware Updates
- View/Edit settings for sending audit data to the Syslog Server
- View/Edit the Local Audit storage warning thresholds
- View/Edit the amount of space allocated Local Audit storage
- Clear/Delete Local Audit records
- View/Edit enabled TLS versions
- View/Edit enabled TLS ciphersuites
- View/Edit X.509 Certificates
- Generate and configure cryptographic keys used to identify the TOE
- Configure cryptographic keys used to authenticate users
- View/Edit the TOE access banner
- View/Edit the session inactivity timeout
- View/Edit user accounts (excluding passwords)
- Set user account passwords
- View/Edit the Pulse One URL

The administrative interfaces provided by the TSF do not allow any of these functions to be accessed by unauthenticated or unauthorized users.

FMT_MOF.1(1)/TrustedUpdate, FMT_MOF.1(2)/TrustedUpdate, FMT_MOF.1(1)/Audit,
FMT_MOF.1(2)/Audit, FMT_MOF.1(1)/AdminAct, FMT_MOF.1/LocSpace, FMT_MTD.1,
FMT_MTD.1/AdminAct, FMT_SMF.1, FMT_SMR.2

7.5 Protection of the TSF

7.5.1 Protection of Administrator Passwords

The TSF does not store plaintext password. The TSF stores the SHA-256 hash of each users' password. Additionally, the TSF does not provide a user interface to view the password hashes.

FPT_APW_EXT.1

7.5.2 Protection of TSF Data (for reading of all symmetric keys)

The TSF stores pre-shared keys, symmetric keys, and private keys in plaintext on the hard disk; however, it does not provide an interface to allow any user to view any of these values.

FPT_SKP_EXT.1

7.5.3 TSF Testing

The TSF performs the following hardware self-tests at power-on:

- Cryptographic library tests:
 - HMAC-SHA-256 integrity check of the library
 - HMAC-SHA-1 KAT
 - HMAC-SHA-256 KAT
 - HMAC-SHA-384 KAT
 - AES 128 ECB Encrypt and Decrypt KAT
 - AES 256 GCM Encrypt and Decrypt KAT
 - RSA 2048 SHA-256 Sign and Verify KAT
 - ECDSA P-224 SHA-512 Sign and Verify PCT
 - DRBG AES-CTR-256 KAT (invoking the instantiate, reseed, and generate functions)
- Firmware checks:
 - RSA 2048 SHA-512 digital signature verification of the manifest file. This file contains a list of all executables that are part of the TSF
 - SHA-256 integrity check of each executable file in the TSF using the pre-calculated hashes from the manifest file.

The Cryptographic library test and the Firmware checks provide a high level of assurance that the firmware has not been tampered with and that the cryptographic algorithms are working properly. The Cryptographic library tests verify that each cryptographic algorithm³ specified in FCS_COP.1 requirements is passing a KAT. The KAT demonstrates that algorithm is functioning properly by invoking the algorithm with hard coded keys and messages and comparing the result to a pre-computed, known to be correct value. The ECDSA PCT shows that the ECDSA algorithm is functioning properly by signing a known value with a known key, and verifying that verifying the computed signature indicates that the signature is valid.

If the cryptographic library tests fail, the TSF will not start up.

If any of the other checks fail, the TSF will log the failure and continue to boot.

FPT_TST_EXT.1

7.5.4 Trusted Update

The TSF allows the Security Administrator to install firmware updates. The Security Administrator obtains candidate updates by downloading them from the Pulse Secure website. When the Security Administrator uploads a firmware update, the TSF performs a RSA 2048 SHA-256 digital signature verification of the update using the Pulse Secure firmware update public key. Pulse Secure retains control over the private key used to sign firmware updates. If the signature check is successful, the TSF installs the update. If the

³ The TSF only tests a single set of parameters for each cryptographic algorithm.

signature check detects tampering with the update and/or signature, the TSF presents the user with an error message and discards the update.

The TSF allows the Security Administrator to view the currently running version of firmware from the System Maintenance > Platform page of the web UI.

FPT_TUD_EXT.1

7.5.5 Reliable Time Stamps

The TOE time function is reliant on the system clock provided by the underlying hardware. The time source is maintained by a reliable hardware clock that is updated by a Security Administrator once a month. The TSF uses system time to timestamp audit log records, to determine user session timeouts, to determine certificate validity, and for HAWK authentication. These uses of time do not require an accuracy finer than one second, and the frequency of updating the time keeps the clock drift under one second.

FPT_STM.1

7.6 TOE Access

The TSF enables Security Administrators to configure an access banner provided with the authentication prompt. The banner can provide warnings against unauthorized access to the TOE as well as any other information that the Security Administrator wishes to communicate.

The TSF presents the access banner prior to authentication when a user connects to the remote web UI or local console CLI described in Section 7.3.2.

User sessions can be terminated by users. The Security Administrator can set the TOE so that local and remote sessions are terminated after a Security Administrator-configured period of inactivity.

FTA_SSL_EXT.1, FTA_SSL.3, FTA_SSL.4, FTA_TAB.1

7.7 Trusted Path/Channels

7.7.1 Inter-TSF Trusted Channel

The TSF communicates with the external syslog server using Syslog over TLS with Authentication as described in Sections 7.1.2 and 7.2.4. The TSF initiates the trusted channel with the Syslog server.

The TSF communicates with the Pulse One management server using HTTPs/TLS (without client authentication as described in Sections 7.2.3 and 7.2.4. The channel with the Pulse One server is authenticated using the HAWK. The TSF initiates the trusted channel with the Pulse One management server. To implement the HAWK authentication scheme, the TSF sends an HMAC-SHA-256 of each message to the Pulse One server. The message includes the current timestamp to prevent replay. The correct HMAC proves the TSF's knowledge of the HAWK secret key.

FTP_ITC.1

7.7.2 Trusted Path

The TSF provides a trusted path for remote administration using HTTPs/TLS as described in Sections 7.2.3 and 7.2.5.

FTP_TRP.1

8. Terms and Definitions

Table 14: TOE Abbreviations and Acronyms	
Abbreviations/ Acronyms	Description
AES	Advanced Encryption Standard
ASCII	American Standard Code for Information Interchange
BIOS	Basic Input/Output System
CBC	Cipher Block Chaining
CLI	Command Line Interface
CMOS	Memory used to store BIOS settings
CRL	Certificate Revocation List
CSR	Certificate Signing Request
DH	Diffie-Hellman
DHE	Diffie-Hellman Ephemeral
DMI	Device Management Interface
DNS	Domain Name System
DSA	Digital Signature Algorithm
ECDH	Elliptic Curve Diffie-Hellman
ECDHE	Elliptic Curve Diffie-Hellman Ephemeral
ECDSA	Elliptic Curve Digital Signature Algorithm
FIPS	Federal Information Processing Standards
GCM	Galois Counter Mode
GUI	Graphical User Interface
HAWK	HTTP Holder of Key
HMAC	Hash Message Authentication Code
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IF-MAP	Interface to Metadata Access Points
IKE	Internet Key Exchange
IP	Internet Protocol
KAT	Known Answer Test
KDF	Key Derivation Function
MB	Megabyte
NCP	Network Communication Protocol
NTP	Network Time Protocol
OCSP	Online Certificate Status Protocol
PCT	Pairwise Consistency Test
PKCS	Public Key Cryptography Standards
RAM	Random Access Memory
RFC	Requests for Comments
RSA	Rivest-Shamir-Adleman
SHA	Secure Hash Algorithm
SNMP	Simple Network Management Protocol
SSH	Secure Shell
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UI	User Interface
VPN	Virtual Private Network

Table 15: CC Abbreviations and Acronyms	
Abbreviations/ Acronyms	Description
CC	Common Criteria
CCRA	Arrangement on the Recognition of Common Criteria Certificates in the field of IT Security
DOD	Department of Defense
NIAP	National Information Assurance Partnership
OSP	Organizational Security Policy
PP	Protection Profile
SAR	Security Assurance Requirement
SFR	Security Functional Requirement
SFP	Security Function Policy
SPD	Security Policy Database
ST	Security Target
TOE	Target of Evaluation
TRRT	Technical Rapid Response Team
TSF	TOE Security Functionality
TSFI	TSF Interface
TSS	TOE Summary Specification

9. References

Table 16: TOE Guidance Documentation			
Reference	Description	Version	Date
[T1]	Pulse Connect Secure Virtual Appliance Operational User Guidance and Preparative Procedures, Pulse Secure LLC	0.4	March 2018

Table 17: Common Criteria v3.1 References			
Reference	Description	Version	Date
[C1]	Common Criteria for Information Technology Security Evaluation Part 1: Introduction and general model CCMB-2012-09-001	V3.1 R4	September 2012
[C2]	Common Criteria for Information Technology Security Evaluation Part 2: Security functional components CCMB-2012-09-002	V3.1 R4	September 2012
[C3]	Common Criteria for Information Technology Security Evaluation Part 3: Security assurance components CCMB-2012-09-003	V3.1 R4	September 2012
[C4]	Common Criteria for Information Technology Security Evaluation Evaluation Methodology CCMB-2012-09-004	V3.1 R4	September 2012

Table 18: Supporting Documentation			
Reference	Description	Version	Date
[NDcPP]	Collaborative Protection Profile for Network Devices	1.0	February 27, 2015
[SD]	Supporting Document Mandatory Technical Document Evaluation Activities for Network Device cPP	1.0	February 27, 2015

Annex A Algorithm Validation Requirements

FCS_CKM.1.1

Note: The following tests require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on factory products.

Key Generation for FIPS PUB 186-4 RSA Schemes

The evaluator shall verify the implementation of RSA Key Generation by the TOE using the Key generation test. This test verifies the ability of the TSF to correctly produce values for the key components including the public verification exponent e , the private prime factors p and q , the public modulus n and the calculation of the private signature exponent d .

Key Pair generation specifies 5 ways (or methods) to generate the primes p and q . These include:

- a) Random Primes:
 - Provable primes
 - Probable primes
- b) Primes with Conditions:
 - Primes p_1, p_2, q_1, q_2, p and q shall all be provable primes
 - Primes $p_1, p_2, q_1,$ and q_2 shall be provable primes and p and q shall be probable primes
 - Primes p_1, p_2, q_1, q_2, p and q shall all be probable primes

To test the key generation method for the Random Provable primes method and for all the Primes with Conditions methods, the evaluator must seed the TSF key generation routine with sufficient data to deterministically generate the RSA key pair. This includes the random seed(s), the public exponent of the RSA key, and the desired key length. For each key length supported, the evaluator shall have the TSF generate 25 key pairs. The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated from a known good implementation.

Key Generation for Elliptic Curve Cryptography (ECC)

FIPS 186-4 ECC Key Generation Test

For each supported NIST curve, i.e., P-256, P-384 and P-521, the evaluator shall require the implementation under test (IUT) to generate 10 private/public key pairs. The private key shall be generated using an approved random bit generator (RBG). To determine correctness, the evaluator shall submit the generated key pairs to the public key verification (PKV) function of a known good implementation.

FIPS 186-4 Public Key Verification (PKV) Test

For each supported NIST curve, i.e., P-256, P-384 and P-521, the evaluator shall generate 10 private/public key pairs using the key generation function of a known good implementation and modify five of the public key values so that they are incorrect, leaving five values unchanged (i.e., correct). The evaluator shall obtain in response a set of 10 PASS/FAIL values.

Key Generation for Finite-Field Cryptography (FFC)

The evaluator shall verify the implementation of the Parameters Generation and the Key Generation for FFC by the TOE using the Parameter Generation and Key Generation test. This test verifies the ability of the TSF to correctly produce values for the field prime p , the cryptographic prime q (dividing $p-1$), the cryptographic group generator g , and the calculation of the private key x and public key y .

The Parameter generation specifies 2 ways (or methods) to generate the cryptographic prime q and the field prime p :

- Primes q and p shall both be provable primes
- Primes q and field prime p shall both be probable primes

and two ways to generate the cryptographic group generator g :

- Generator g constructed through a verifiable process
- Generator g constructed through an unverifiable process.

The Key generation specifies 2 ways to generate the private key x :

- $\text{len}(q)$ bit output of RBG where $1 \leq x \leq q-1$
- $\text{len}(q) + 64$ bit output of RBG, followed by a mod $q-1$ operation where $1 \leq x \leq q-1$.

The security strength of the RBG must be at least that of the security offered by the FFC parameter set.

To test the cryptographic and field prime generation method for the provable primes method and/or the group generator g for a verifiable process, the evaluator must seed the TSF parameter generation routine with sufficient data to deterministically generate the parameter set.

For each key length supported, the evaluator shall have the TSF generate 25 parameter sets and key pairs. The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated from a known good implementation. Verification must also confirm:

- $g \neq 0, 1$
- q divides $p-1$
- $g^q \bmod p = 1$
- $g^x \bmod p = y$

for each FFC parameter set and key pair.

FCS_CKM.2.1

Key Establishment Schemes

The evaluator shall verify the implementation of the key establishment schemes of the supported by the TOE using the applicable tests below.

SP800-56A Key Establishment Schemes

The evaluator shall verify a TOE's implementation of SP800-56A key agreement schemes using the following Function and Validity tests. These validation tests for each key agreement scheme verify that a TOE has implemented the components of the key agreement scheme according to the specifications in the Recommendation. These components include the calculation of the DLC primitives (the shared secret value Z) and the calculation of the derived keying material (DKM) via the Key Derivation Function (KDF). If key confirmation is supported, the evaluator shall also verify that the components of key confirmation have been implemented correctly, using the test procedures described below. This includes the parsing of the DKM, the generation of MACdata and the calculation of MACtag.

Function Test

The Function test verifies the ability of the TOE to implement the key agreement schemes correctly. To conduct this test the evaluator shall generate or obtain test vectors from a known good implementation of the TOE supported schemes. For each supported key agreement scheme-key agreement role combination, KDF type, and, if supported, key confirmation role-key confirmation type combination, the

tester shall generate 10 sets of test vectors. The data set consists of one set of domain parameter values (FFC) or the NIST approved curve (ECC) per 10 sets of public keys. These keys are static, ephemeral or both depending on the scheme being tested.

The evaluator shall obtain the DKM, the corresponding TOE's public keys (static and/or ephemeral), the MAC tag(s), and any inputs used in the KDF, such as the Other Information field OI and TOE id fields.

If the TOE does not use a KDF defined in SP 800-56A, the evaluator shall obtain only the public keys and the hashed value of the shared secret.

The evaluator shall verify the correctness of the TSF's implementation of a given scheme by using a known good implementation to calculate the shared secret value, derive the keying material DKM, and compare hashes or MACtags generated from these values.

If key confirmation is supported, the TSF shall perform the above for each implemented approved MAC algorithm.

Validity Test

The Validity test verifies the ability of the TOE to recognize another party's valid and invalid key agreement results with or without key confirmation. To conduct this test, the evaluator shall obtain a list of the supporting cryptographic functions included in the SP800-56A key agreement implementation to determine which errors the TOE should be able to recognize. The evaluator generates a set of 24 (FFC) or 30 (ECC) test vectors consisting of data sets including domain parameter values or NIST approved curves, the evaluator's public keys, the TOE's public/private key pairs, MACTag, and any inputs used in the KDF, such as the other info and TOE id fields.

The evaluator shall inject an error in some of the test vectors to test that the TOE recognizes invalid key agreement results caused by the following fields being incorrect: the shared secret value Z, the DKM, the other information field OI, the data to be MACed, or the generated MACTag. If the TOE contains the full or partial (only ECC) public key validation, the evaluator will also individually inject errors in both parties' static public keys, both parties' ephemeral public keys and the TOE's static private key to assure the TOE detects errors in the public key validation function and/or the partial key validation function (in ECC only). At least two of the test vectors shall remain unmodified and therefore should result in valid key agreement results (they should pass).

The TOE shall use these modified test vectors to emulate the key agreement scheme using the corresponding parameters. The evaluator shall compare the TOE's results with the results using a known good implementation verifying that the TOE detects these errors.

SP800-56B Key Establishment Scheme Testing

The evaluator shall verify that the TSS describes whether the TOE acts as a sender, a recipient, or both for RSA-based key establishment schemes.

If the TOE acts as a sender, the following assurance activity shall be performed to ensure the proper operation of every TOE supported combination of RSA-based key establishment scheme:

- a) To conduct this test the evaluator shall generate or obtain test vectors from a known good implementation of the TOE supported schemes. For each combination of supported key establishment scheme and its options (with or without key confirmation if supported, for each supported key confirmation MAC function if key confirmation is supported, and for each supported mask generation function if KTS-OAEP is supported), the tester shall generate 10 sets of test vectors. Each test vector shall include the RSA public key, the plaintext keying material, any

additional input parameters if applicable, the MacKey and MacTag if key confirmation is incorporated, and the outputted ciphertext. For each test vector, the evaluator shall perform a key establishment encryption operation on the TOE with the same inputs (in cases where key confirmation is incorporated, the test shall use the MacKey from the test vector instead of the randomly generated MacKey used in normal operation) and ensure that the outputted ciphertext is equivalent to the ciphertext in the test vector.

If the TOE acts as a receiver, the following assurance activities shall be performed to ensure the proper operation of every TOE supported combination of RSA-based key establishment scheme:

- a) To conduct this test the evaluator shall generate or obtain test vectors from a known good implementation of the TOE supported schemes. For each combination of supported key establishment scheme and its options (with or without key confirmation if supported, for each supported key confirmation MAC function if key confirmation is supported, and for each supported mask generation function if KTS-OAEP is supported), the tester shall generate 10 sets of test vectors. Each test vector shall include the RSA private key, the plaintext keying material (KeyData), any additional input parameters if applicable, the MacTag in cases where key confirmation is incorporated, and the outputted ciphertext. For each test vector, the evaluator shall perform the key establishment decryption operation on the TOE and ensure that the outputted plaintext keying material (KeyData) is equivalent to the plaintext keying material in the test vector. In cases where key confirmation is incorporated, the evaluator shall perform the key confirmation steps and ensure that the outputted MacTag is equivalent to the MacTag in the test vector.
- b) The evaluator shall ensure that the TSS describes how the TOE handles decryption errors. In accordance with NIST Special Publication 800-56B, the TOE must not reveal the particular error that occurred, either through the contents of any outputted or logged error message or through timing variations. If KTS-OAEP is supported, the evaluator shall create separate contrived ciphertext values that trigger each of the three decryption error checks described in NIST Special Publication 800-56B section 7.2.2.3, ensure that each decryption attempt results in an error, and ensure that any outputted or logged error message is identical for each. If KTS-KEM-KWS is supported, the evaluator shall create separate contrived ciphertext values that trigger each of the three decryption error checks described in NIST Special Publication 800-56B section 7.2.3.3, ensure that each decryption attempt results in an error, and ensure that any outputted or logged error message is identical for each.

FCS_COP.1.1(1)

AES-CBC Known Answer Tests

There are four Known Answer Tests (KATs), described below. In all KATs, the plaintext, ciphertext, and IV values shall be 128-bit blocks. The results from each test may either be obtained by the evaluator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

KAT-1. To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of 10 plaintext values and obtain the ciphertext value that results from AES-CBC encryption of the given plaintext using a key value of all zeros and an IV of all zeros. Five plaintext values shall be encrypted with a 128-bit all-zeros key, and the other five shall be encrypted with a 256-bit all-zeros key.

To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using 10 ciphertext values as input and AES-CBC decryption.

KAT-2. To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of 10 key values and obtain the ciphertext value that results from AES-CBC encryption of an all-zeros plaintext using the given key value and an IV of all zeros. Five of the keys shall be 128-bit keys, and the other five shall be 256-bit keys

To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using an all-zero ciphertext value as input and AES-CBC decryption.

KAT-3. To test the encrypt functionality of AES-CBC, the evaluator shall supply the two sets of key values described below and obtain the ciphertext value that results from AES encryption of an all-zeros plaintext using the given key value and an IV of all zeros. The first set of keys shall have 128 128-bit keys, and the second set shall have 256 256-bit keys. Key I in each set shall have the leftmost i bits be ones and the rightmost $N-i$ bits be zeros, for I in $[1, N]$.

To test the decrypt functionality of AES-CBC, the evaluator shall supply the two sets of key and ciphertext value pairs described below and obtain the plaintext value that results from AES-CBC decryption of the given ciphertext using the given key and an IV of all zeros. The first set of key/ciphertext pairs shall have 128 128-bit key/ciphertext pairs, and the second set of key/ciphertext pairs shall have 256 256-bit key/ciphertext pairs. Key i in each set shall have the leftmost i bits be ones and the rightmost $N-i$ bits be zeros, for i in $[1, N]$. The ciphertext value in each pair shall be the value that results in an all-zeros plaintext when decrypted with its corresponding key.

KAT-4. To test the encrypt functionality of AES-CBC, the evaluator shall supply the set of 128 plaintext values described below and obtain the two ciphertext values that result from AES-CBC encryption of the given plaintext using a 128-bit key value of all zeros with an IV of all zeros and using a 256-bit key value of all zeros with an IV of all zeros, respectively. Plaintext value i in each set shall have the leftmost i bits be ones and the rightmost $128-i$ bits be zeros, for i in $[1, 128]$.

To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using ciphertext values of the same form as the plaintext in the encrypt test as input and AES-CBC decryption.

AES-CBC Multi-Block Message Test

The evaluator shall test the encrypt functionality by encrypting an i -block message where $1 < i \leq 10$. The evaluator shall choose a key, an IV and plaintext message of length i blocks and encrypt the message, using the mode to be tested, with the chosen key and IV. The ciphertext shall be compared to the result of encrypting the same plaintext message with the same key and IV using a known good implementation.

The evaluator shall also test the decrypt functionality for each mode by decrypting an i -block message where $1 < i \leq 10$. The evaluator shall choose a key, an IV and a ciphertext message of length i blocks and decrypt the message, using the mode to be tested, with the chosen key and IV. The plaintext shall be compared to the result of decrypting the same ciphertext message with the same key and IV using a known good implementation.

AES-CBC Monte Carlo Tests

The evaluator shall test the encrypt functionality using a set of 200 plaintext, IV, and key 3-tuples. 100 of these shall use 128 bit keys, and 100 shall use 256 bit keys. The plaintext and IV values shall be 128-bit blocks. For each 3-tuple, 1000 iterations shall be run as follows:

```
# Input: PT, IV, Key
for i = 1 to 1000:
    if i == 1:
        CT[1] = AES-CBC-Encrypt(Key, IV, PT)
        PT = IV
    else:
        CT[i] = AES-CBC-Encrypt(Key, PT)
        PT = CT[i-1]
```

The ciphertext computed in the 1000th iteration (i.e., CT[1000]) is the result for that trial. This result shall be compared to the result of running 1000 iterations with the same values using a known good implementation.

The evaluator shall test the decrypt functionality using the same test as for encrypt, exchanging CT and PT and replacing AES-CBC-Encrypt with AES-CBC-Decrypt.

AES-GCM Test

The evaluator shall test the authenticated encrypt functionality of AES-GCM for each combination of the following input parameter lengths:

128 bit and 256 bit keys

- a) Two plaintext lengths. One of the plaintext lengths shall be a non-zero integer multiple of 128 bits, if supported. The other plaintext length shall not be an integer multiple of 128 bits, if supported.
- b) Three AAD lengths. One AAD length shall be 0, if supported. One AAD length shall be a non-zero integer multiple of 128 bits, if supported. One AAD length shall not be an integer multiple of 128 bits, if supported.
- c) Two IV lengths. If 96 bit IV is supported, 96 bits shall be one of the two IV lengths tested.

The evaluator shall test the encrypt functionality using a set of 10 key, plaintext, AAD, and IV tuples for each combination of parameter lengths above and obtain the ciphertext value and tag that results from AES-GCM authenticated encrypt. Each supported tag length shall be tested at least once per set of 10. The IV value may be supplied by the evaluator or the implementation being tested, as long as it is known

The evaluator shall test the decrypt functionality using a set of 10 key, ciphertext, tag, AAD, and IV 5-tuples for each combination of parameter lengths above and obtain a Pass/Fail result on authentication and the decrypted plaintext if Pass. The set shall include five tuples that Pass and five that Fail.

The results from each test may either be obtained by the evaluator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

FCS_COP.1.1(2)

ECDSA Algorithm Tests

ECDSA FIPS 186-4 Signature Generation Test

For each supported NIST curve (i.e., P-256, P-384 and P-521) and SHA function pair, the evaluator shall generate 10 1024-bit long messages and obtain for each message a public key and the resulting signature values R and S. To determine correctness, the evaluator shall use the signature verification function of a known good implementation.

ECDSA FIPS 186-4 Signature Verification Test

For each supported NIST curve (i.e., P-256, P-384 and P-521) and SHA function pair, the evaluator shall generate a set of 10 1024-bit message, public key and signature tuples and modify one of the values (message, public key or signature) in five of the 10 tuples. The evaluator shall obtain in response a set of 10 PASS/FAIL values.

RSA Signature Algorithm Tests

Signature Generation Test

The evaluator shall verify the implementation of RSA Signature Generation by the TOE using the Signature Generation Test. To conduct this test the evaluator must generate or obtain 10 messages from a trusted reference implementation for each modulus size/SHA combination supported by the TSF. The evaluator shall have the TOE use their private key and modulus value to sign these messages.

The evaluator shall verify the correctness of the TSF's signature using a known good implementation and the associated public keys to verify the signatures.

Signature Verification Test

The evaluator shall perform the Signature Verification test to verify the ability of the TOE to recognize another party's valid and invalid signatures. The evaluator shall inject errors into the test vectors produced during the Signature Verification Test by introducing errors in some of the public keys e , messages, IR format, and/or signatures. The TOE attempts to verify the signatures and returns success or failure.

The evaluator shall use these test vectors to emulate the signature verification test using the corresponding parameters and verify that the TOE detects these errors.

FCS_COP.1.1(3)

The TSF hashing functions can be implemented in one of two modes. The first mode is the byte-oriented mode. In this mode the TSF only hashes messages that are an integral number of bytes in length; i.e., the length (in bits) of the message to be hashed is divisible by 8. The second mode is the bit-oriented mode. In this mode the TSF hashes messages of arbitrary length. As there are different tests for each mode, an indication is given in the following sections for the bit-oriented vs. the byte-oriented testmacs.

The evaluator shall perform all of the following tests for each hash algorithm implemented by the TSF and used to satisfy the requirements of this PP.

Short Messages Test - Bit-oriented Mode

The evaluators devise an input set consisting of $m+1$ messages, where m is the block length of the hash algorithm. The length of the messages range sequentially from 0 to m bits. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Short Messages Test - Byte-oriented Mode

The evaluators devise an input set consisting of $m/8+1$ messages, where m is the block length of the hash algorithm. The length of the messages range sequentially from 0 to $m/8$ bytes, with each message being an integral number of bytes. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Selected Long Messages Test - Bit-oriented Mode

The evaluators devise an input set consisting of m messages, where m is the block length of the hash algorithm (e.g. 512 bits for SHA-256). The length of the i th message is $m + 99*i$, where $1 \leq i \leq m$. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Selected Long Messages Test - Byte-oriented Mode

The evaluators devise an input set consisting of $m/8$ messages, where m is the block length of the hash algorithm (e.g. 512 bits for SHA-256). The length of the i th message is $m + 8*99*i$, where $1 \leq i \leq m/8$. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Pseudorandomly Generated Messages Test

This test is for byte-oriented implementations only. The evaluators randomly generate a seed that is n bits long, where n is the length of the message digest produced by the hash function to be tested. The evaluators then formulate a set of 100 messages and associated digests by following the algorithm provided in Figure 1 of [SHAVS]. The evaluators then ensure that the correct result is produced when the messages are provided to the TSF.