Cisco Systems, Inc.

Email Security Appliance running AsyncOS 13.0

Assurance Activity Report

Version 1.4

July 29, 2021

Document prepared by

Lightship Security

www.lightshipsec.com

Table of Contents

1 Introduction

1 This Assurance Activity Report (AAR) documents the evaluation activities performed by Lightship Security for the evaluation identified in Table 1. The AAR is produced in accordance with National Information Assurance Program (NIAP) reporting guidelines.

1.1 Evaluation Identifiers

Table 1: Evaluation Identifiers

Scheme	Canadian Common Criteria Scheme
Evaluation Facility	Lightship Security
Developer/Sponsor	Cisco Systems, Inc.
TOE	Email Security Appliance running AsyncOS 13.0
Security Target	Cisco Email Security Appliance Common Criteria Security Target, 1.3, 29 July 2021
Protection Profile	collaborative Protection Profile for Network Devices, v2.1 (CPP_ND), 24-Sept-2018

1.2 Evaluation Methods

2 The evaluation was performed using the methods and standards identified in Table 2.

Table 2: Evaluation Methods

Evaluation Criteria	CC v3.1R5	
Evaluation Methodology	CEM v3.1R5	
Supporting Documents	Supporting Document Mandatory Technical Document Evaluation Activities for Network Device cPP, version 2.1, September-2018	
Interpretations	**TID**	**Technical Decision**
	TD0395	NIT Technical Decision for Different Handling of TLS1.1 and TLS1.2
	TD0396	NIT Technical Decision for FCS_TLSC_EXT.1.1, Test 2 **N/A since TLSC not claimed.**

TD0397	NIT Technical Decision for Fixing AES-CTR Mode Tests
TD0398	NIT Technical Decision for FCS_SSH*EXT.1.1 RFCs for AES-CTR
TD0399	NIT Technical Decision for Manual installation of CRL (FIA_X509_EXT.2)
TD0400	NIT Technical Decision for FCS_CKM.2 and elliptic curve-based key establishment
TD0401	NIT Technical Decision for Reliance on external servers to meet SFRs
TD0402	NIT Technical Decision for RSA-based FCS_CKM.2 Selection
TD0407	NIT Technical Decision for handling Certification of Cloud Deployments
TD0408	NIT Technical Decision for local vs. remote administrator accounts
TD0409	NIT decision for Applicability of FIA_AFL.1 to key-based SSH authentication
TD0410	NIT technical decision for Redundant assurance activities associated with FAU_GEN.1
TD0411	NIT Technical Decision for FCS_SSHC_EXT.1.5, Test 1 - Server and client side seem to be confused
TD0412	NIT Technical Decision for FCS_SSHS_EXT.1.5 SFR and AA discrepancy
TD0423	NIT Technical Decision for Clarification about application of RfI#201726rev2
TD0424	NIT Technical Decision for NDcPP v2.1 Clarification - FCS_SSHC/S_EXT1.5

	TD0425	NIT Technical Decision for Cut-and-paste Error for Guidance AA
	TD0447	NIT Technical Decision for Using 'diffie-hellman-group-exchange-sha256' in FCS_SSHC/S_EXT.1.7
	TD0450	NIT Technical Decision for RSA-based ciphers and the Server Key Exchange message
	TD0451	NIT Technical Decision for ITT Comm UUID Reference Identifier
	TD0453	NIT Technical Decision for Clarify authentication methods SSH clients can use to authenticate SSH se
	TD0475	NIT Technical Decision for Separate traffic consideration for SSH rekey
	TD0477	NIT Technical Decision for Clarifying FPT_TUD_EXT.1 Trusted Update
	TD0478	NIT Technical Decision for Application Notes for FIA_X509_EXT.1 iterations
	TD0480	NIT Technical Decision for Granularity of audit events
	TD0481	NIT Technical Decision for FCS_(D)TLSC_EXT.X.2 IP addresses in reference identifiers **N/A since FCS_(D)TLSC_EXT.* not claimed.**
	TD0482	NIT Technical Decision for Identification of usage of cryptographic schemes
	TD0483	NIT Technical Decision for Applicability of FPT_APW_EXT.1
	TD0484	NIT Technical Decision for Interactive sessions in FTA_SSL_EXT.1 & FTA_SSL.3

	TD0528	NIT Technical Decision for Missing EAs for FCS_NTP_EXT.1.4 **N/A since NTP not claimed.**
	TD0529	NIT Technical Decision for OCSP and Authority Information Access extension **N/A since OCSP not claimed.**
	TD0530	NIT Technical Decision for FCS_TLSC_EXT.1.1 5e test clarification **N/A since TLSC not claimed.**
	TD0531	NIT Technical Decision for Challenge-Response for Authentication
	TD0532	NIT Technical Decision for Use of seeds with higher entropy
	TD0533	NIT Technical Decision for FTP_ITC.1 with signed downloads
	TD0535	NIT Technical Decision for Clarification about digital signature algorithms for FTP_TUD.1
	TD0536	NIT Technical Decision for Update Verification Inconsistency
	TD0538	NIT Technical Decision for Outdated link to allowed-with list
	TD0547	NIT Technical Decision for Clarification on developer disclosure of AVA_VAN
	TD0570	NiT Technical Decision for Clarification about FIA_AFL.1
	TD0571	NiT Technical Decision for Guidance on how to handle FIA_AFL.1
	TD0572	NiT Technical Decision for Restricting FTP_ITC.1 to only IP address identifiers

2 TOE Details

2.1 Overview

1 The Cisco Email Security Appliance is a network device. ESA is an appliance that provides comprehensive email protection services for a company's email system. It is an email protection product that monitors Simple Mail Transfer Protocol (SMTP) network traffic, analyzes the monitored network traffic using various techniques, and reacts to identified threats associated with email messages (such as spam and inappropriate or malicious content).

2.2 TOE Models/Platforms

2 The TOE is a hardware and software solution that makes up the Cisco ESA. The TOE hardware is comprised of the following: C190, C195, C390, C395, C690, C690X, C695, and C695F. Additionally, the C100v, C300v, C600v virtual appliances running on Cisco UCS servers with ESXi 6.0 are included as TOEs. The TOE software is the ESA AsyncOS software version 13.0. For a complete list of models and hardware platforms, please refer to section 1.5 of the Security Target.

2.3 Reference Documents

Table 3: List of Reference Documents

Ref	Document
[ST]	Cisco Email Security Appliance Common Criteria Security Target, v1.3, 29 July 2021
[SUPP]	Cisco Email Security Appliance running AsyncOS 13.0 Common Criteria Operational User Guidance and Preparative Procedures, v1.1, 29 July 2021
[ADMIN]	User Guide for AsyncOS 13.0 for Cisco Email Security Appliances – GD (General Deployment), Last modified: 2020-02-03
[CLI]	CLI Reference Guide for AsyncOS 13.0 for Cisco Email Security Appliances - GD(General Deployment), Last modified: 2020-02-03
[HW170]	Cisco 170 Series Hardware Installation Guide, Text Part Number: OL-28365-01
[HWx95]	Cisco Email Security Appliance C195, C395, C695, and C695F Getting Started Guide, June 7, 2019
[VAPP]	Cisco Content Security Virtual Appliance Installation Guide, March 22, 2021

2.4 Summary of SFRs

Table 4: List of SFRs

Requirement	Title
FAU_GEN.1	Audit Data Generation
FAU_GEN.2	User Identity Association
FAU_STG_EXT.1	Protected Audit Event Storage
FCS_CKM.1	Cryptographic Key Generation
FCS_CKM.2	Cryptographic Key Establishment
FCS_CKM.4	Cryptographic Key Destruction
FCS_COP.1/DataEncryption	Cryptographic Operation (AES Data Encryption/Decryption)
FCS_COP.1/SigGen	Cryptographic Operation (Signature Generation and Verification)
FCS_COP.1/Hash	Cryptographic Operation (Hash Algorithm)
FCS_COP.1/KeyedHash	Cryptographic Operation (Keyed Hash Algorithm)
FCS_HTTPS_EXT.1	HTTPS Protocol
FCS_RBG_EXT.1	Random Bit Generation
FCS_SSHC_EXT.1	SSH Client Protocol
FCS_SSHS_EXT.1	SSH Server Protocol
FCS_TLSS_EXT.1	TLS Server Protocol
FIA_AFL.1	Authentication Failure Management
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/Rev	X.509 Certificate Validation
FIA_X509_EXT.2	X.509 Certificate Authentication
FIA_X509_EXT.3	X.509 Certificate Requests
FMT_MOF.1/Functions	Management of security functions behaviour
FMT_MOF.1/ManualUpdate	Management of security functions behaviour
FMT_MTD.1/CoreData	Management of TSF Data

Requirement	Title
FMT_MTD.1/CryptoKeys	Management of TSF Data
FMT_SMF.1	Specification of Management Functions
FMT_SMR.2	Restrictions on Security Roles
FPT_APW_EXT.1	Protection of Administrator Passwords
FPT_SKP_EXT.1	Protection of TSF Data (for reading of all pre-shared, symmetric and private keys)
FPT_STM_EXT.1	Reliable Time Stamps
FPT_TST_EXT.1	TSF testing
FPT_TUD_EXT.1	Trusted update
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
FTP_ITC.1	Inter-TSF trusted channel
FTP_TRP.1/Admin	Trusted Path

3 Evaluation Activities for SFRs

3.1 Security Audit (FAU)

3.1.1 FAU_GEN.1 Audit data generation

3.1.1.1 TSS

3 For the administrative task of generating/import of, changing, or deleting of cryptographic keys as defined in FAU_GEN.1.1c, the TSS should identify what information is logged to identify the relevant key.

Findings:	In section 6.1 of the [ST] under FAU_GEN.1, the TSS states that the TOE logs a reference to the associated key.

4 For distributed TOEs the evaluator shall examine the TSS to ensure that it describes which of the overall required auditable events defined in FAU_GEN.1.1 are generated and recorded by which TOE components. The evaluator shall ensure that this mapping of audit events to TOE components accounts for, and is consistent with, information provided in Table 1, as well as events in Tables 2, 4, and 5 (where applicable to the overall TOE). This includes that the evaluator shall confirm that all components defined as generating audit information for a particular SFR should also contribute to that SFR as defined in the mapping of SFRs to TOE components, and that the audit records generated by each component cover all the SFRs that it implements.

Findings:	The TOE is not a distributed TOE.

3.1.1.2 Guidance Documentation

5 **TD410 -** The evaluator shall check the guidance documentation and ensure that it provides an example of each auditable event required by FAU_GEN.1 (i.e. at least one instance of each auditable event – comprising the mandatory, optional and selection-based SFR sections as applicable – shall be provided from the actual audit record).

Findings:	Section 5 of [SUPP] provides an example of each of the relevant auditable events.

6 The evaluator shall also make a determination of the administrative actions related to TSF data related to configuration changes. The evaluator shall examine the guidance documentation and make a determination of which administrative commands, including subcommands, scripts, and configuration files, are related to the configuration (including enabling or disabling) of the mechanisms implemented in the TOE that are necessary to enforce the requirements specified in the cPP. The evaluator shall document the methodology or approach taken while determining which actions in the administrative guide are related to TSF data related to configuration changes. The evaluator may perform this activity as part of the activities associated with ensuring that the corresponding guidance documentation satisfies the requirements related to it.

Findings:	The evaluator performed this activity as part of those AAs associated with ensuring the corresponding guidance documentation satisfied their independent requirements. However, overall, the evaluator considered the administrator guides published by the vendor, including the [SUPP]. The evaluator reviewed the contents of the

	documentation and looked specifically for functionality related to the scope of the evaluation. Where there was missing or incomplete descriptions for the functionality such that the user could not complete the testing AAs, the evaluator requested the vendor to supply augmented guidance information.

3.1.1.3 Tests

7 The evaluator shall test the TOE's ability to correctly generate audit records by having the TOE generate audit records for the events listed in the table of audit events and administrative actions listed above. This should include all instances of an event: for instance, if there are several different I&A mechanisms for a system, the FIA_UIA_EXT.1 events must be generated for each mechanism. The evaluator shall test that audit records are generated for the establishment and termination of a channel for each of the cryptographic protocols contained in the ST. If HTTPS is implemented, the test demonstrating the establishment and termination of a TLS session can be combined with the test for an HTTPS session. When verifying the test results, the evaluator shall ensure the audit records generated during testing match the format specified in the guidance documentation, and that the fields in each audit record have the proper entries.

Findings:	These tests are conducted throughout the test plan.

8 For distributed TOEs the evaluator shall perform tests on all TOE components according to the mapping of auditable events to TOE components in the Security Target. For all events involving more than one TOE component when an audit event is triggered, the evaluator has to check that the event has been audited on both sides (e.g. failure of building up a secure communication channel between the two components). This is not limited to error cases but includes also events about successful actions like successful build up/tear down of a secure communication channel between TOE components.

Findings:	The TOE is not a distributed TOE.

9 Note that the testing here can be accomplished in conjunction with the testing of the security mechanisms directly.

3.1.2 FAU_GEN.2 User identity association

10 TSS & Guidance Documentation

11 The TSS and Guidance Documentation requirements for FAU_GEN.2 are already covered by the TSS and Guidance Documentation requirements for FAU_GEN.1.

3.1.2.1 Tests

12 This activity should be accomplished in conjunction with the testing of FAU_GEN.1.1.

13 For distributed TOEs the evaluator shall verify that where auditable events are instigated by another component, the component that records the event associates the event with the identity of the instigator. The evaluator shall perform at least one test on one component where another component instigates an auditable event. The evaluator shall verify that the event is recorded by the component as expected and the event is associated with the instigating component. It is assumed that an event instigated by another component can at least be generated for building up a secure

channel between two TOE components. If for some reason (could be e.g. TSS or Guidance Documentation) the evaluator would come to the conclusion that the overall TOE does not generate any events instigated by other components, then this requirement shall be omitted.

Findings:	The TOE is not a distributed TOE.

3.1.3 FAU_STG_EXT.1 Protected audit event storage

3.1.3.1 TSS

14 The evaluator shall examine the TSS to ensure it describes the means by which the audit data are transferred to the external audit server, and how the trusted channel is provided.

Findings:	The [ST] in section 6.1 under FAU_STG_EXT.1 indicates that the TOE can be configured to deliver logs to a remote server using SCP (SSH).

15 The evaluator shall examine the TSS to ensure it describes the amount of audit data that are stored locally; what happens when the local audit data store is full; and how these records are protected against unauthorized access.

Findings:	In section 6.1 in the [ST], under FAU_STG_EXT.1, the TSS states that the TOE can store a configurable number of logs up to a configurable size limit. If the space available for storing audit records is exhausted, the TOE will start to overwrite the oldest records. Only Authorized Administrators can clear the local logs, and there is no TOE interface that allows for administrators to modify the contents of the local audit records.

16 The evaluator shall examine the TSS to ensure it describes whether the TOE is a standalone TOE that stores audit data locally or a distributed TOE that stores audit data locally on each TOE component or a distributed TOE that contains TOE components that cannot store audit data locally on themselves but need to transfer audit data to other TOE components that can store audit data locally. The evaluator shall examine the TSS to ensure that for distributed TOEs it contains a list of TOE components that store audit data locally. The evaluator shall examine the TSS to ensure that for distributed TOEs that contain components which do not store audit data locally but transmit their generated audit data to other components it contains a mapping between the transmitting and storing TOE components.

Findings:	The TOE is not a distributed TOE.

17 The evaluator shall examine the TSS to ensure that it details the behaviour of the TOE when the storage space for audit data is full. When the option ‘overwrite previous audit record’ is selected this description should include an outline of the rule for overwriting audit data. If ‘other actions’ are chosen such as sending the new audit data to an external IT entity, then the related behaviour of the TOE shall also be detailed in the TSS.

Findings:	As in the previous work units, section 6.1 of the [ST] states that if the space available for storing audit records is exhausted, the TOE will start to overwrite the oldest records.

18 The evaluator shall examine the TSS to ensure that it details whether the transmission of audit information to an external IT entity can be done in real-time or periodically. In case the TOE does not perform transmission in real-time the evaluator

needs to verify that the TSS provides details about what event stimulates the transmission to be made as well as the possible as well as acceptable frequency for the transfer of audit data.

Findings:	Section 6.1 of the [ST] under FAU_STG_EXT.1 indicates that the transmission is done based on configurable time and space rules provided by the administrator.

19 For distributed TOEs the evaluator shall examine the TSS to ensure it describes to which TOE components this SFR applies and how audit data transfer to the external audit server is implemented among the different TOE components (e.g. every TOE components does its own transfer or the data is sent to another TOE component for central transfer of all audit events to the external audit server).

Findings:	The TOE is not a distributed TOE.

20 For distributed TOEs the evaluator shall examine the TSS to ensure it describes which TOE components are storing audit information locally and which components are buffering audit information and forwarding the information to another TOE component for local storage. For every component the TSS shall describe the behaviour when local storage space or buffer space is exhausted.

Findings:	The TOE is not a distributed TOE.

3.1.3.2 Guidance Documentation

21 The evaluator shall also examine the guidance documentation to ensure it describes how to establish the trusted channel to the audit server, as well as describe any requirements on the audit server (particular audit server protocol, version of the protocol required, etc.), as well as configuration of the TOE needed to communicate with the audit server.

Findings:	Section 3.4 of the [SUPP] provides instructions for enabling the remote offloading of the logs. This is done using the CLI (using the “logconfig” command). The same section of [SUPP] stipulates that “…the SCP server on a remote syslog server [uses] SCP over SSHv2 to ensure the session is secured.”

22 The evaluator shall also examine the guidance documentation to determine that it describes the relationship between the local audit data and the audit data that are sent to the audit log server. For example, when an audit event is generated, is it simultaneously sent to the external server and the local store, or is the local store used as a buffer and “cleared” periodically by sending the data to the audit server.

Findings:	Section 3.4 of the [SUPP] provides the necessary instructions to an admin that the configuration of the remote log push relies on the admin setting rules about frequency and log space. It is not real time. The [SUPP] in section 3.4 also says: “In the evaluated configuration, the rollover time must be used and must be set for the shortest allotted time, so the records are as close to simultaneously being pushed to the remote syslog server.” During testing, it was found that the shortest time that can be set is 60 seconds.

23 The evaluator shall also ensure that the guidance documentation describes all possible configuration options for FAU_STG_EXT.1.3 and the resulting behaviour of the TOE for each possible configuration. The description of possible configuration options and resulting behaviour shall correspond to those described in the TSS.

Findings:	Section 3.4 of the [SUPP] provides an appropriate level of information on the fact that admins can set both frequency- and log volume-based rules to push to the remote

	server. This is consistent with the information found in section 6.1 of the [ST] under FAU_STG_EXT.1.

3.1.3.3 Tests

24 Testing of the trusted channel mechanism for audit will be performed as specified in the associated assurance activities for the particular trusted channel mechanism. The evaluator shall perform the following additional tests for this requirement:

a) Test 1: The evaluator shall establish a session between the TOE and the audit server according to the configuration guidance provided. The evaluator shall then examine the traffic that passes between the audit server and the TOE during several activities of the evaluator’s choice designed to generate audit data to be transferred to the audit server. The evaluator shall observe that these data are not able to be viewed in the clear during this transfer, and that they are successfully received by the audit server. The evaluator shall record the particular software (name, version) used on the audit server during testing. The evaluator shall verify that the TOE is capable of transferring audit data to an external audit server automatically without administrator intervention.

Note	Verification that the data is encrypted is satisfied by FTP_ITC.1 for the logging channel. The logging server uses SSHv2 described in the Test Setup. Due to the log-forwarding mechanism used on logging server, the audit records are therefore confirmed to have been successfully received by the audit server whenever the test cases are run.

b) Test 2: The evaluator shall perform operations that generate audit data and verify that this data is stored locally. The evaluator shall perform operations that generate audit data until the local storage space is exceeded and verifies that the TOE complies with the behaviour defined in FAU_STG_EXT.1.3. Depending on the configuration this means that the evaluator has to check the content of the audit data when the audit data is just filled to the maximum and then verifies that

1) The audit data remains unchanged with every new auditable event that should be tracked but that the audit data is recorded again after the local storage for audit data is cleared (for the option ‘drop new audit data’ in FAU_STG_EXT.1.3).

2) The existing audit data is overwritten with every new auditable event that should be tracked according to the specified rule (for the option ‘overwrite previous audit records’ in FAU_STG_EXT.1.3)

3) The TOE behaves as specified (for the option ‘other action’ in FAU_STG_EXT.1.3).

High-Level Test Description
Configure a log subscription with a rollover capability. Show that the logs rollover when the configured thresholds are reached and overwrite the previous records.
PASS

c) Test 3: If the TOE complies with FAU_STG_EXT.2/LocSpace the evaluator shall verify that the numbers provided by the TOE according to the selection for FAU_STG_EXT.2/LocSpace are correct when performing the tests for FAU_STG_EXT.1.3

Findings:	The TOE does not claim this function.

d) Test 4: For distributed TOEs, Test 1 defined above should be applicable to all TOE components that forward audit data to an external audit server. For the local storage according to FAU_STG_EXT.1.2 and FAU_STG_EXT.1.3 the Test 2 specified above shall be applied to all TOE components that store audit data locally. For all TOE components that store audit data locally and comply with FAU_STG_EXT.2/LocSpace Test 3 specified above shall be applied. The evaluator shall verify that the transfer of audit data to an external audit server is implemented.

Findings:	The TOE is not a distributed TOE.

3.2 Cryptographic Support (FCS)

3.2.1 FCS_CKM.1 Cryptographic Key Generation

3.2.1.1 TSS

25 The evaluator shall ensure that the TSS identifies the key sizes supported by the TOE. If the ST specifies more than one scheme, the evaluator shall examine the TSS to verify that it identifies the usage for each scheme.

Findings:	This information can be found in section 6.1 of the [ST] under FCS_CKM.1 and FCS_CKM.2. Specifically, the TOE is capable of generating 2048-bit (or greater) RSA keys for use in a CSR and for use in the SSH hostkey. The TOE also acts as a sender and receiver for Diffie-Hellman and EC Diffie-Hellman key establishment and therefore is responsible for key generation for those key establishment algorithms. The [ST] in section 6.1 for FCS_CKM.1 and FCS_CKM.2 provides a sub-table showing how each scheme is used. This sub-table is consistent with the claims and the testing.

3.2.1.2 Guidance Documentation

26 The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected key generation scheme(s) and key size(s) for all cryptographic protocols defined in the Security Target.

Findings:	Operational key generation is configured for TLS. Section 3.3.4 of the [SUPP] indicates that TLS RSA private keys can be generated via the “certconfig” CLI command. SSH hostkeys for the TOE are generated at software installation time and cannot be regenerated through standard operational means. The [SUPP] indicates in section 3.2.1 that it is necessary to operate the device in FIPS mode to automatically configure the FIPS approved algorithms and key sizes.

3.2.1.3 Tests

27 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. Generation of long-term cryptographic keys (i.e. keys that are not ephemeral keys/session keys) might be performed automatically (e.g. during initial start-up). Testing of key generation must cover not only administrator invoked key generation but also automated key generation (if supported).

Key Generation for FIPS PUB 186-4 RSA Schemes

28 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*.

29 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 p1, p2, q1,q2, p and q shall all be provable primes
- Primes p1, p2, q1, and q2 shall be provable primes and p and q shall be probable primes
- Primes p1, p2, q1,q2, p and q shall all be probable primes

30 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.

Findings:	RSA key generation is covered by the following CAVP certificates all of which claim RSA KeyGen 186-4 for 2048-bit RSA keys: A397, A402, A405, A406, C924, C905. These claims are consistent with FCS_CKM.1 in the [ST] section 5.2.2.

Key Generation for Elliptic Curve Cryptography (ECC)

FIPS 186-4 ECC Key Generation Test

31 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

32 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.

Findings: ECDSA key generation is covered by the following CAVP certificates all of which claim ECDSA KeyGen 186-4 for NIST curves P-256, P-384 and P-521: A397, A402, A405, A406, C924, C905. These claims are consistent with FCS_CKM.1 in the [ST] section 5.2.2.

Key Generation for Finite-Field Cryptography (FFC)

33 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.

34 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

35 and two ways to generate the cryptographic group generator g:

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

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

- len(q) bit output of RBG where 1 <=x <= q-1
- len(q) + 64 bit output of RBG, followed by a mod q-1 operation and a +1 operation, where 1<= x<=q-1.

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

38 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.

39 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 != 0,1
- q divides p-1
- g^q mod p = 1
- g^x mod p = y

40 for each FFC parameter set and key pair.

Findings: FFC EC key generation is covered by the following CAVP certificates all of which claim KAS ECC Component for NIST curves P-256, P-384 and P-521: A397, A402, A405, A406, C924, C905. These claims are consistent with FCS_CKM.1 and FCS_CKM.2 in the [ST] section 5.2.2. Diffie-hellman group 14 is covered by additional testing below.

Diffie-Hellman Group 14

41 Testing for FFC Schemes using Diffie-Hellman group 14 is done as part of testing in CKM.2.1.

3.2.2 FCS_CKM.2 Cryptographic Key Establishment

3.2.2.1 TSS

42 **TD0482 -** The evaluator shall ensure that the supported key establishment schemes correspond to the key generation schemes identified in FCS_CKM.1.1. If the ST specifies more than one scheme, the evaluator shall examine the TSS to verify that it identifies the usage for each scheme. It is sufficient to provide the scheme, SFR, and service in the TSS.

43 If Diffie-Hellman group 14 is selected from FCS_CKM.2.1, the TSS shall affirm that the TOE implements RFC 3526 Section 3.

44 The intent of this activity is to be able to identify the scheme being used by each service. This would mean, for example, one way to document scheme usage could be:

Scheme	SFR	Service
RSA	FCS_TLSS_EXT.1	Administration
ECDH	FCS_SSHC_EXT.1	Audit Server
Diffie-Hellman (group 14)	FCS_SSHC_EXT.1	Backup Server
ECDH	FCS_IPSEC_EXT.1	Authentication Server

45 The information provided in the example above does not necessarily have to be included as a table but can be presented in other ways as long as the necessary data is available.

Findings: The [ST] in section 6.1 for FCS_CKM.1 and FCS_CKM.2 provides a sub-table showing how each scheme is used. This sub-table is consistent with the claims and the testing.

3.2.2.2 Guidance Documentation

46 The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected key establishment scheme(s).

Findings: Operational key establishment configuration is possible for both SSH and TLS. Section 3.3.2 of the [SUPP] indicates that SSH key exchange algorithms can be configured using the “sshconfig” CLI command. Section 3.3.3 of the [SUPP] provides the “sslconfig” CLI command to configure the TLS ciphers (which encode the key establishment algorithms as part of the ciphersuite specification).

The [SUPP] indicates in section 3.2.1 that it is necessary to operate the device in FIPS mode to automatically configure the FIPS approved algorithms and key sizes.

3.2.2.3 Tests

Key Establishment Schemes

47 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

48 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

49 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.

50 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.

51 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.

52 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 MAC tags generated from these values.

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

Validity Test

54 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.

55 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).

56 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.

Findings:	FFC EC key generation is covered by the following CAVP certificates all of which claim KAS ECC Component for NIST curves P-256, P-384 and P-521: A397, A402, A405, A406, C924, C905. These claims are consistent with FCS_CKM.1 and FCS_CKM.2 in the [ST] section 5.2.2.

RSA-based key establishment schemes

57 **TD0402 -** The evaluator shall verify the correctness of the TSF's implementation of RSAES-PKCS1-v1_5 by using a known good implementation for each protocol selected in FTP_TRP.1/Admin, FTP_TRP.1/Join, FTP_ITC.1 and FPT_ITT.1 that uses RSAES-PKCS1-v1_5.

High-Level Test Description
FCS_TLSS_EXT.1.3: Using a Lightship developed TLS client, connect to the TOE using a valid pure RSA ciphersuite and verify that the certificate that comes back from the Server Certificate message matches the expected bit size.
PASS

Diffie-Hellman Group 14

58 The evaluator shall verify the correctness of the TSF's implementation of Diffie-Hellman group 14 by using a known good implementation for each protocol selected in FTP_TRP.1/Admin, FTP_TRP.1/Join, FTP_ITC.1 and FPT_ITT.1 that uses Diffie-Hellman group 14.

High-Level Test Description
FCS_SSHS_EXT.1.7 Test 2: Using an independent SSH client, forcibly negotiate each of the claimed key exchange algorithms in turn and show that it results in a successful connection (the TOE claims group 14).
PASS

3.2.3 FCS_CKM.4 Cryptographic Key Destruction

3.2.3.1 TSS

59 The evaluator examines the TSS to ensure it lists all relevant keys (describing the origin and storage location of each), all relevant key destruction situations (e.g. factory reset or device wipe function, disconnection of trusted channels, key change as part of a secure channel protocol), and the destruction method used in each case. For the purpose of this Evaluation Activity the relevant keys are those keys that are relied upon to support any of the SFRs in the Security Target. The evaluator confirms that the description of keys and storage locations is consistent with the functions carried out by the TOE (e.g. that all keys for the TOE-specific secure channels and protocols, or that support FPT_APW.EXT.1 and FPT_SKP_EXT.1, are accounted for[1]). In particular, if a TOE claims not to store plaintext keys in non-volatile memory then the evaluator checks that this is consistent with the operation of the TOE.

Findings:	Section 6.1 of the [ST] provides information on how keys and CSPs are zeroized. Additional information is provided in section 7 of the [ST] in the form of a table outlining the keys, their storage means and how they are zeroized. The types of keys and CSPs are consistent with the claims made in section 5 of the [ST].

60 The evaluator shall check to ensure the TSS identifies how the TOE destroys keys stored as plaintext in non-volatile memory, and that the description includes identification and description of the interfaces that the TOE uses to destroy keys (e.g., file system APIs, key store APIs).

Findings:	The table included in section 7 of the [ST] indicates that the keys are overwritten with specific values using the FOM to overwrite memory directly. For other keys, they are overwritten when a new key is generated. As per the FCS_CKM.4 SFR in section 5 of the [ST], the TOE uses an abstraction to refer to the key.

61 Note that where selections involve '*destruction of reference*' (for volatile memory) or '*invocation of an interface'* (for non-volatile memory) then the relevant interface definition is examined by the evaluator to ensure that the interface supports the selection(s) and description in the TSS. In the case of non-volatile memory the evaluator includes in their examination the relevant interface description for each media type on which plaintext keys are stored. The presence of OS-level and storage device-level swap and cache files is not examined in the current version of the Evaluation Activity.

Findings:	The [ST] in section 5 claims that the TOE uses an interface provided by a part of the TSF. However, the interfaces are not exposed via the user-serviceable CLI or Web UI and are done automatically by the cryptographic module.

62 Where the TSS identifies keys that are stored in a non-plaintext form, the evaluator shall check that the TSS identifies the encryption method and the key-encrypting-key used, and that the key-encrypting-key is either itself stored in an encrypted form or that it is destroyed by a method included under FCS_CKM.4.

Findings:	N/A – Section 7 in the [ST] only lists keys stored in plaintext form.

63 The evaluator shall check that the TSS identifies any configurations or circumstances that may not conform to the key destruction requirement (see further discussion in

[1] Where keys are stored encrypted or wrapped under another key then this may need to be explained in order to allow the evaluator to confirm the consistency of the description of keys with the TOE functions.

the Guidance Documentation section below). Note that reference may be made to the Guidance Documentation for description of the detail of such cases where destruction may be prevented or delayed.

Findings:	The [ST] TSS identifies no configurations or circumstances that may not conform to the key destruction requirement.

64 Where the ST specifies the use of “a value that does not contain any CSP” to overwrite keys, the evaluator examines the TSS to ensure that it describes how that pattern is obtained and used, and that this justifies the claim that the pattern does not contain any CSPs.

Findings:	N/A - The use of “*a value that does not contain any CSP*” is not included in the ST.

3.2.3.2 Guidance Documentation

65 A TOE may be subject to situations that could prevent or delay key destruction in some cases. The evaluator shall check that the guidance documentation identifies configurations or circumstances that may not strictly conform to the key destruction requirement, and that this description is consistent with the relevant parts of the TSS (and any other supporting information used). The evaluator shall check that the guidance documentation provides guidance on situations where key destruction may be delayed at the physical layer.

66 For example, when the TOE does not have full access to the physical memory, it is possible that the storage may be implementing wear-levelling and garbage collection. This may result in additional copies of the key that are logically inaccessible but persist physically. Where available, the TOE might then describe use of the TRIM command[2] and garbage collection to destroy these persistent copies upon their deletion (this would be explained in TSS and Operational Guidance).

Findings:	Section 3.3.7.1 of the [SUPP] provides information where key material might be present in coredump files which are often generated as a result of error conditions. Such coredump files can be overwritten with zeros using the “wipedata” CLI command.

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

3.2.4.1 Tests

AES-CBC Known Answer Tests

67 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.

[2] Where TRIM is used then the TSS and/or guidance documentation is also expected to describe how the keys are stored such that they are not inaccessible to TRIM, (e.g. they would need not to be contained in a file less than 982 bytes which would be completely contained in the master file table).

68 **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.

69 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.

70 **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.

71 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.

72 **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].

73 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.

74 **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].

75 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

76 The evaluator shall test the encrypt functionality by encrypting an *i*-block message where 1 < *i* <=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.

77 The evaluator shall also test the decrypt functionality for each mode by decrypting an *i*-block message where 1 < *i* <=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

78 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]
```

79 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.

80 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.

Findings:	AES encryption and decryption is covered by the following CAVP certificates all of which claim AES with key sizes of 128- and 256-bits: A397, A402, A405, A406, C924, C905. These claims are consistent with FCS_COP.1/DataEncryption in the [ST] section 5.2.2.

AES-GCM Test

81 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.

a. **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.

b. **Two IV lengths**. If 96 bit IV is supported, 96 bits shall be one of the two IV lengths tested.

82 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.

83 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.

84 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.

Findings:	AES-GCM mode is not claimed.

AES-CTR Known Answer Tests

85 **TD0397 –** The counter (CTR) mode is a confidentiality mode that features the application of the forward cipher to a set of input blocks, called counters, to produce a sequence of output blocks that are exclusive-ORed with the plaintext to produce the ciphertext, and vice versa. Due to the fact that Counter Mode does not specify the counter that is used, it is not possible to implement an automated test for this mode. The generation and management of the counter is tested through FCS_SSH*_EXT.1.4. If CBC and/or GCM are selected in FCS_COP.1/DataEncryption, the test activities for those modes sufficiently demonstrate the correctness of the AES algorithm. If CTR is the only selection in FCS_COP.1/DataEncryption, the AES-CBC Known Answer Test, AES-GCM Known Answer Test, or the following test shall be performed (all of these tests demonstrate the correctness of the AES algorithm):

86 There are four Known Answer Tests (KATs) described below to test a basic AES encryption operation (AES-ECB mode). For all KATs, the plaintext, ~~IV,~~ and ciphertext values shall be 128-bit blocks. The results from each test may either be obtained by the validator 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.

87 KAT-1 To test the encrypt functionality, the evaluator shall supply a set of 5 plaintext values for each selected keysize and obtain the ciphertext value that results from encryption of the given plaintext using a key value of all zeros.

88 KAT-2 To test the encrypt functionality, the evaluator shall supply a set of 5 key values for each selected keysize and obtain the ciphertext value that results from encryption of an all zeros plaintext using the given key value.

89 KAT-3 To test the encrypt functionality, the evaluator shall supply a set of key values for each selected keysize as described below and obtain the ciphertext values that result from AES encryption of an all zeros plaintext using the given key values. A set of 128 128-bit keys, a set of 192 192-bit keys, and/or a set of 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].

90 KAT-4 To test the encrypt functionality, the evaluator shall supply the set of 128 plaintext values described below and obtain the ciphertext values that result from encryption of the given plaintext using each selected keysize with a key value of all zeros (e.g. 256 ciphertext values will be generated if 128 bits and 256 bits are

selected and 384 ciphertext values will be generated if all keysizes are selected). Plaintext value i in each set shall have the leftmost bits be ones and the rightmost 128-i bits be zeros, for i in [1, 128]

AES-CTR Multi-Block Message Test

91 The evaluator shall test the encrypt functionality by encrypting an i-block message where 1 less-than i less-than-or-equal to 10 (test shall be performed using AES-ECB mode). For each i the evaluator shall choose a key and plaintext message of length i blocks and encrypt the message, using the mode to be tested, with the chosen key. The ciphertext shall be compared to the result of encrypting the same plaintext message with the same key using a known good implementation. The evaluator shall perform this test using each selected keysize.

AES-CTR Monte-Carlo Test

92 The evaluator shall test the encrypt functionality using 100 plaintext/key pairs. The plaintext values shall be 128-bit blocks. For each pair, 1000 iterations shall be run as follows:

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

93 The ciphertext computed in the 1000th iteration 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 perform this test using each selected keysize.

Findings:	AES-CTR encryption and decryption is covered by the following CAVP certificates all of which claim AES with key sizes of 128- and 256-bits: A397, A402, A405, A406, C924, C905. These claims are consistent with FCS_COP.1/DataEncryption in the [ST] section 5.2.2.

3.2.5 FCS_COP.1/SigGen Cryptographic Operation (Signature Generation and Verification

3.2.5.1 Tests

ECDSA Algorithm Tests

ECDSA FIPS 186-4 Signature Generation Test

94 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.

Findings:	The TOE does not claim ECDSA signature generation.

ECDSA FIPS 186-4 Signature Verification Test

95 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.

Findings:	The TOE does not claim ECDSA signature verification.

RSA Signature Algorithm Tests

Signature Generation Test

96 The evaluator generates or obtains 10 messages for each modulus size/SHA combination supported by the TOE. The TOE generates and returns the corresponding signatures.

97 The evaluator shall verify the correctness of the TOE's signature using a trusted reference implementation of the signature verification algorithm and the associated public keys to verify the signatures.

Findings:	RSA Signature Generation is covered by the following CAVP certificates all of which claim RSA SigGen 186-4 using PKCS 1.5 and 2048-bit RSA keys: A397, A402, A405, A406, C924, C905. These claims are consistent with FCS_COP.1/SigGen in the [ST] section 5.2.2.

Signature Verification Test

98 For each modulus size/hash algorithm selected, the evaluator generates a modulus and three associated key pairs, (*d*, *e*). Each private key *d* is used to sign six pseudorandom messages each of 1024 bits using a trusted reference implementation of the signature generation algorithm. Some of the public keys, *e*, messages, or signatures are altered so that signature verification should fail. For both the set of original messages and the set of altered messages: the modulus, hash algorithm, public key *e* values, messages, and signatures are forwarded to the TOE, which then attempts to verify the signatures and returns the verification results.

99 The evaluator verifies that the TOE confirms correct signatures on the original messages and detects the errors introduced in the altered messages.

Findings:	RSA Signature Verification is covered by the following CAVP certificates all of which claim RSA SigVer 186-4 using PKCS 1.5 and 2048-bit RSA keys: A397, A402, A405, A406, C924, C905. These claims are consistent with FCS_COP.1/SigGen in the [ST] section 5.2.2.

3.2.6 FCS_COP.1/Hash Cryptographic Operation (Hash Algorithm)

3.2.6.1 TSS

100 The evaluator shall check that the association of the hash function with other TSF cryptographic functions (for example, the digital signature verification function) is documented in the TSS.

Findings:	In the [ST] in section 6.1, SHA and SHA2 are claimed to be used in TLS, SSH, firmware image hash verification and signature verification of X.509 certificates.

3.2.6.2 Guidance Documentation

101 The evaluator checks the AGD documents to determine that any configuration that is required to configure the required hash sizes is present.

Findings:	Section 3.3.2 of the [SUPP] provides an administrator a means to configure the HMAC for TOE server-side SSH protocol using the “sshconfig” CLI command. Section 3.3.4 of the [SUPP] permits the administrator to configure the set of TLS ciphersuites that can be used by the TOE for the web interface using the “sslconfig” CLI command. In both cases, the [SUPP] provides the set of permitted values that meet the evaluated configuration. The [SUPP] indicates in section 3.2.1 that it is necessary to operate the device in FIPS mode to automatically configure the FIPS approved algorithms and key sizes.

3.2.6.3 Tests

102 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.

103 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

104 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

105 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

106 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 ith message is m + 99*i, where 1 ≤ i ≤ 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

107 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 ith message is m + 8*99*i, where 1 ≤ i ≤ 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

108 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.

Findings:	Hashing is covered by the following CAVP certificates all of which claim SHA1 and SHA2-256, SHA2-384 and SHA2-512: A397, A402, A405, A406, C924, C905. These claims are consistent with FCS_COP.1/Hash in the [ST] section 5.2.2.

3.2.7 FCS_COP.1/KeyedHash Cryptographic Operation (Keyed Hash Algorithm)

3.2.7.1 TSS

109 The evaluator shall examine the TSS to ensure that it specifies the following values used by the HMAC function: key length, hash function used, block size, and output MAC length used.

Findings:	The [ST] in section 6.1 for the table entry FCS_COP.1/Hash, FCS_COP.1/KeyedHash indicates that the TOE uses HMAC-SHA1. The key length is 160-bits, hash function is SHA1, block size is 512 bits and output MAC length is 160 bits.

3.2.7.2 Tests

110 For each of the supported parameter sets, the evaluator shall compose 15 sets of test data. Each set shall consist of a key and message data. The evaluator shall have the TSF generate HMAC tags for these sets of test data. The resulting MAC tags shall be compared to the result of generating HMAC tags with the same key and message data using a known good implementation.

Findings:	HMAC is covered by the following CAVP certificates all of which claims HMAC-SHA1: A397, A402, A405, A406, C924, C905. These claims are consistent with FCS_COP.1/KeyedHash in the [ST] section 5.2.2.

3.2.8 FCS_RBG_EXT.1 Extended: Cryptographic Operation (Random Bit Generation)

111 Documentation shall be produced—and the evaluator shall perform the activities—in accordance with Appendix D of [NDcPP].

3.2.8.1 TSS

112 The evaluator shall examine the TSS to determine that it specifies the DRBG type, identifies the entropy source(s) seeding the DRBG, and state the assumed or calculated min-entropy supplied either separately by each source or the min-entropy contained in the combined seed value.

Findings: According to section 6.1 of the [ST] under FCS_RBG_EXT.1, "The TOE implements a NIST-approved AES-CTR Deterministic Random Bit Generator (DRBG), as specified in SP 800-90 seeded by an entropy source that accumulates entropy from a TSF-software based noise source as described in FCS_RBG_EXT.1. This output is used directly to seed the DRBG." Furthermore, the "…deterministic RBG is seeded with a minimum of 256 bits of entropy."

3.2.8.2 Guidance Documentation

113 The evaluator shall confirm that the guidance documentation contains appropriate instructions for configuring the RNG functionality.

Findings: The [SUPP] indicates in section 3.2.1 that it is necessary to operate the device in FIPS mode to automatically configure the FIPS approved algorithms and key sizes.

3.2.8.3 Tests

114 The evaluator shall perform 15 trials for the RNG implementation. If the RNG is configurable, the evaluator shall perform 15 trials for each configuration.

115 If the RNG has prediction resistance enabled, each trial consists of (1) instantiate DRBG, (2) generate the first block of random bits (3) generate a second block of random bits (4) uninstantiate. The evaluator verifies that the second block of random bits is the expected value. The evaluator shall generate eight input values for each trial. The first is a count (0 – 14). The next three are entropy input, nonce, and personalization string for the instantiate operation. The next two are additional input and entropy input for the first call to generate. The final two are additional input and entropy input for the second call to generate. These values are randomly generated. "generate one block of random bits" means to generate random bits with number of returned bits equal to the Output Block Length (as defined in NIST SP800-90A).

116 If the RNG does not have prediction resistance, each trial consists of (1) instantiate DRBG, (2) generate the first block of random bits (3) reseed, (4) generate a second block of random bits (5) uninstantiate. The evaluator verifies that the second block of random bits is the expected value. The evaluator shall generate eight input values for each trial. The first is a count (0 – 14). The next three are entropy input, nonce, and personalization string for the instantiate operation. The fifth value is additional input to the first call to generate. The sixth and seventh are additional input and entropy input to the call to reseed. The final value is additional input to the second generate call.

117 The following paragraphs contain more information on some of the input values to be generated/selected by the evaluator.

Entropy input: the length of the entropy input value must equal the seed length.

Nonce: If a nonce is supported (CTR_DRBG with no Derivation Function does not use a nonce), the nonce bit length is one-half the seed length.

Personalization string: The length of the personalization string must be <= seed length. If the implementation only supports one personalization string length, then the same length can be used for both values. If more than one string length is support, the evaluator shall use personalization strings of two different lengths. If the implementation does not use a personalization string, no value needs to be supplied.

Additional input: the additional input bit lengths have the same defaults and restrictions as the personalization string lengths.

Findings:	AES CTR-mode DRBG is covered by the following CAVP certificates which all claim use of a 256-bit key: A397, A402, A405, A406, C924, C905. These claims are consistent with FCS_RBG_EXT.1 in the [ST] section 5.2.2.

3.3 Identification and Authentication (FIA)

3.3.1 FIA_AFL.1 Authentication Failure Management

3.3.1.1 TSS

118 The evaluator shall examine the TSS to determine that it contains a description, for each supported method for remote administrative actions, of how successive unsuccessful authentication attempts are detected and tracked. The TSS shall also describe the method by which the remote administrator is prevented from successfully logging on to the TOE, and the actions necessary to restore this ability.

Findings:	As per section 6.1 of the [ST] under FIA_AFL.1: “When the Authorized Administrator attempting to log into the administrative CLI or GUI interface reaches the administratively set maximum number of failed authentication attempts, the user will not be granted access to the administrative functionality of the TOE until an Authorized Administrator resets the user's number of failed login attempts through the administrative CLI using the “userconfig” command or GUI Edit User webpage.”

119 The evaluator shall examine the TSS to confirm that the TOE ensures that authentication failures by remote administrators cannot lead to a situation where no administrator access is available, either permanently or temporarily (e.g. by providing local logon which is not subject to blocking).

Findings:	Section 6.1 of the [ST] under FIA_AFL.1 informs the reader that the “admin” account is not subject to the lock out at the local console. This is to ensure the administrators do not get totally locked out of the TOE.

3.3.1.2 Guidance Documentation

120 The evaluator shall examine the guidance documentation to ensure that instructions for configuring the number of successive unsuccessful authentication attempts and time period (if implemented) are provided, and that the process of allowing the remote administrator to once again successfully log on is described for each “action” specified (if that option is chosen). If different actions or mechanisms are implemented depending on the secure protocol employed (e.g., TLS vs. SSH), all must be described.

Findings:	Section 4.5.1 of the [SUPP] provides information to the administrator to configure the lockout parameters and to unlock the users. The only action available to the administrator is to manually unlock users. Unlocking users can be done using the CLI (“userconfig”) or the GUI.

121 The evaluator shall examine the guidance documentation to confirm that it describes, and identifies the importance of, any actions that are required in order to ensure that administrator access will always be maintained, even if remote administration is made permanently or temporarily unavailable due to blocking of accounts as a result of FIA_AFL.1.

Findings:	Section 4.5.1 of the [SUPP] informs administrators that the “admin” user is not susceptible to lockout at the local console and can be used to rescue the system if necessary.

3.3.1.3 Tests

122 The evaluator shall perform the following tests for each method by which remote administrators access the TOE (e.g. any passwords entered as part of establishing the connection protocol or the remote administrator application):

a. Test 1: The evaluator shall use the operational guidance to configure the number of successive unsuccessful authentication attempts allowed by the TOE (and, if the time period selection in FIA_AFL.1.2 is included in the ST, then the evaluator shall also use the operational guidance to configure the time period after which access is re-enabled). The evaluator shall test that once the authentication attempts limit is reached, authentication attempts with valid credentials are no longer successful.

High-Level Test Description
Configure the account locking mechanism to 5 attempts. Log in with 4 bad attempts and then on the 5th attempt, log in successfully and show it is not locked out. Then lock out the user with 5 bad attempts and show that the 6th is denied even if the proper password is used. Using the administrator account, unlock the locked user and then show that the user can log in again. Repeat the above for both SSH and Web UI interfaces. Show that users cannot be locked out at the local console.
PASS

b. Test 2: After reaching the limit for unsuccessful authentication attempts as in Test 1 above, the evaluator shall proceed as follows.

   If the administrator action selection in FIA_AFL.1.2 is included in the ST then the evaluator shall confirm by testing that following the operational guidance and performing each action specified in the ST to re-enable the remote administrator's access results in successful access (when using valid credentials for that administrator).

   If the time period selection in FIA_AFL.1.2 is included in the ST then the evaluator shall wait for just less than the time period configured in Test 1 and show that an authorisation attempt using valid credentials does not result in successful access. The evaluator shall then wait until just after the time period configured in Test 1 and show that an authorisation attempt using valid credentials results in successful access.

Note:	See previous test case.

3.3.2 FIA_PMG_EXT.1 Password Management

3.3.2.1 Guidance Documentation

123 The evaluator shall examine the guidance documentation to determine that it:

a. identifies the characters that may be used in passwords and provides guidance to security administrators on the composition of strong passwords, and

b. provides instructions on setting the minimum password length and describes the valid minimum password lengths supported.

Findings:	Section 4.3 of the [SUPP] offers the administrator information on the characters that can be used to compose a password as well as the available optional password complexity rules that the product has to offer. Section 4.3 of the [SUPP] indicates that the minimum password length can be set using the Web UI. The available range of values for the minimum password length are also provided in the [SUPP] in section 4.3 but informs the admin that a minimum length of 15 is required for the purposes of the evaluated configuration.

3.3.2.2 Tests

124 The evaluator shall perform the following tests.

a. Test 1: The evaluator shall compose passwords that either meet the requirements, or fail to meet the requirements, in some way. For each password, the evaluator shall verify that the TOE supports the password. While the evaluator is not required (nor is it feasible) to test all possible compositions of passwords, the evaluator shall ensure that all characters, and a minimum length listed in the requirement are supported, and justify the subset of those characters chosen for testing.

High-Level Test Description
Change the password length to be 15 characters using supported characters. Change the password for the built-in ‘admin’ user using the identified TSFI. Show that the password can be used to login to the Web GUI, SSH and local console. Change the password length to be 8 characters. Change the password for the built-in ‘admin’ back to a known good password. Using the admin account, change the password for another user to be only 7 characters and show it is rejected. Change the password for another user to be 8 characters and show it is accepted. Repeat this on the CLI and Web UI.
PASS

3.3.3 FIA_UIA_EXT.1 User Identification and Authentication

3.3.3.1 TSS

125 The evaluator shall examine the TSS to determine that it describes the logon process for each logon method (local, remote (HTTPS, SSH, etc.)) supported for the product. This description shall contain information pertaining to the credentials allowed/used, any protocol transactions that take place, and what constitutes a “successful logon”.

Findings: Section 6.1 of the [ST] under heading FIA_UIA_EXT.1,FIA_UAU_EXT.2 provides the necessary information about how the TOE processes logging into the TOE. The TOE offers a local console CLI secured by a username and password, a remote CLI secured using either username/password or username/public key over SSH, and a remote web-based GUI operating over HTTPS secured using a username and password.

126 The evaluator shall examine the TSS to determine that it describes which actions are allowed before user identification and authentication. The description shall cover authentication and identification for local and remote TOE administration.

Findings: Section 6.1 of the [ST] under heading FIA_UIA_EXT.1,FIA_UAU_EXT.2 indicates “[t]he TOE requires all users to be successfully identified and authenticated before allowing any TSF mediated actions to be performed, except for the login banner that is displayed prior to user authentication.” This is consistent with the claims made in section 5.2 of the [ST].

127 For distributed TOEs the evaluator shall examine that the TSS details how Security Administrators are authenticated and identified by all TOE components. If not all TOE components support authentication of Security Administrators according to FIA_UIA_EXT.1 and FIA_UAU_EXT.2, the TSS shall describe how the overall TOE functionality is split between TOE components including how it is ensured that no unauthorized access to any TOE component can occur.

Findings: The TOE is not a distributed TOE.

128 For distributed TOEs, the evaluator shall examine the TSS to determine that it describes for each TOE component which actions are allowed before user identification and authentication. The description shall cover authentication and identification for local and remote TOE administration. For each TOE component that does not support authentication of Security Administrators according to FIA_UIA_EXT.1 and FIA_UAU_EXT.2 the TSS shall describe any unauthenticated services/services that are supported by the component.

Findings: The TOE is not a distributed TOE.

3.3.3.2 Guidance Documentation

129 The evaluator shall examine the guidance documentation to determine that any necessary preparatory steps (e.g., establishing credential material such as pre-shared keys, tunnels, certificates, etc.) to logging in are described. For each supported the login method, the evaluator shall ensure the guidance documentation provides clear instructions for successfully logging on. If configuration is necessary to ensure the services provided before login are limited, the evaluator shall determine that the guidance documentation provides sufficient instruction on limiting the allowed services.

Findings: Information about how to log onto the TOE over the remote SSH CLI can be found in [CLI] in section “Accessing the Command Line Interface (CLI)”. This information pertains to both physical and virtual images. While that section references use of telnet, the [SUPP] in section 3.3 informs the end-user not to use telnet.

Logging onto the device using the remote Web UI interface is provided in [ADMIN] in chapter 2 (“Accessing the Appliance”). This information is applicable to both physical and virtual devices.

Local console access for physical appliances described in [HW170] can be found in the section titled “Connecting the Interface Cables and Verifying Connectivity”. In the [HWx95] document, this information can be found in the section titled “Log In to the Appliance Using the CLI”. Local console requirements about baud rate, etc. are described.

The local console use for virtual appliances for ESXi is briefly described in [VAPP] in section “Deploy on VMWare ESXi”. Further information about the ESXi hypervisor and how to access the console of a virtual device would be available from VMWare.

3.3.3.3 Tests

130 The evaluator shall perform the following tests for each method by which administrators access the TOE (local and remote), as well as for each type of credential supported by the login method:

a. Test 1: The evaluator shall use the guidance documentation to configure the appropriate credential supported for the login method. For that credential/login method, the evaluator shall show that providing correct I&A information results in the ability to access the system, while providing incorrect information results in denial of access.

High-Level Test Description
Log into the identified management interface using a known-good credential and logout. Login into the identified management interface using a known-bad credential and logout. Ensure the appropriate audit messages appear.
PASS

b. Test 2: The evaluator shall configure the services allowed (if any) according to the guidance documentation, and then determine the services available to an external remote entity. The evaluator shall determine that the list of services available is limited to those specified in the requirement.

High-Level Test Description
The device does not have any services configured prior to I&A. All claimed services available to remote entities are identified as part of AVA_VAN.1 test scanning.
PASS

c. Test 3: For local access, the evaluator shall determine what services are available to a local administrator prior to logging in, and make sure this list is consistent with the requirement.

High-Level Test Description
The device does not have any services configured prior to I&A. All claimed services available to local entities are identified as part of AVA_VAN.1 test scanning.
PASS

d. Test 4: For distributed TOEs where not all TOE components support the authentication of Security Administrators according to FIA_UIA_EXT.1 and FIA_UAU_EXT.2, the evaluator shall test that the components authenticate Security Administrators as described in the TSS.

Findings:	The TOE is not a distributed TOE.

3.3.4 FIA_UAU_EXT.2 Password-based Authentication Mechanism

131 Evaluation Activities for this requirement are covered under those for FIA_UIA_EXT.1. If other authentication mechanisms are specified, the evaluator shall include those methods in the activities for FIA_UIA_EXT.1.

3.3.5 FIA_UAU.7 Protected Authentication Feedback

3.3.5.1 Tests

132 The evaluator shall perform the following test for each method of local login allowed:

a. Test 1: The evaluator shall locally authenticate to the TOE. While making this attempt, the evaluator shall verify that at most obscured feedback is provided while entering the authentication information.

High-Level Test Description
Log into the local management interface. Ensure the password field does not echo characters – even a masking character -- as claimed by the ST.
PASS

3.4 Security management (FMT)

3.4.1 General requirements for distributed TOEs

3.4.1.1 TSS

133 For distributed TOEs it is required to verify the TSS to ensure that it describes how every function related to security management is realized for every TOE component and shared between different TOE components. The evaluator shall confirm that all relevant aspects of each TOE component are covered by the FMT SFRs.

Findings: The TOE is not a distributed TOE.

3.4.1.2 Guidance Documentation

134 For distributed TOEs it is required to verify the Guidance Documentation to describe management of each TOE component. The evaluator shall confirm that all relevant aspects of each TOE component are covered by the FMT SFRs.

Findings: The TOE is not a distributed TOE.

3.4.1.3 Tests

135 Tests defined to verify the correct implementation of security management functions shall be performed for every TOE component. For security management functions that are implemented centrally, sampling should be applied when defining the evaluator’s tests (ensuring that all components are covered by the sample).

Findings: The TOE is not a distributed TOE.

3.4.2 FMT_MOF.1/ManualUpdate

3.4.2.1 TSS

136 For distributed TOEs see chapter 2.4.1.1. There are no specific requirements for non-distributed TOEs.

Findings: The TOE is not a distributed TOE.

3.4.2.2 Guidance Documentation

137 The evaluator shall examine the guidance documentation to determine that any necessary steps to perform manual update are described. The guidance documentation shall also provide warnings regarding functions that may cease to operate during the update (if applicable).

Findings: The information has been found in section 2.1 of the [SUPP].

138 For distributed TOEs the guidance documentation shall describe all steps how to update all TOE components. This shall contain description of the order in which components need to be updated if the order is relevant to the update process. The guidance documentation shall also provide warnings regarding functions of TOE

components and the overall TOE that may cease to operate during the update (if applicable).

Findings:	The TOE is not a distributed TOE.

3.4.2.3 Tests

139 The evaluator shall try to perform the update using a legitimate update image without prior authentication as security administrator (either by authentication as a user with no administrator privileges or without user authentication at all – depending on the configuration of the TOE). The attempt to update the TOE shall fail.

High-Level Test Description
Log into the Web GUI using an account with privileges which should not permit upgrades. Attempt to upgrade the device. The action should fail. Log into the CLI using an account with privileges which should not permit upgrades. Attempt to upgrade the device. The action should fail.
PASS

140 The evaluator shall try to perform the update with prior authentication as security administrator using a legitimate update image. This attempt should be successful. This test case should be covered by the tests for FPT_TUD_EXT.1 already.

Findings:	This test case is performed as part of FPT_TUD_EXT.1.

3.4.3 FMT_MTD.1/CoreData Management of TSF Data

3.4.3.1 TSS

141 The evaluator shall examine the TSS to determine that, for each administrative function identified in the guidance documentation; those that are accessible through an interface prior to administrator log-in are identified. For each of these functions, the evaluator shall also confirm that the TSS details how the ability to manipulate the TSF data through these interfaces is disallowed for non-administrative users.

Findings:	The [ST] in section 6.1 (repeated throughout, such as in FIA_UIA_EXT.1, FMT_MTD.1/CoreData, etc) consistently indicates that no in-scope administrative functionality is available prior to login by an administrator. Note that the TOE banner can be displayed to end-users before establishing a login session as per FIA_UIA_EXT.1.

142 If TOE supports handling of X.509v3 certificates and implements a trust store, the evaluator shall examine the TSS to determine that it contains sufficient information to describe how the ability to manage the TOE's trust store is restricted.

Findings:	The [ST] in section 6.1 under heading FMT_MTD.1/CoreData states that only Authorized Administrators are capable of "uploading and enabling X509 certificates". The term "Authorized Administrator" is defined in the same section to denote any administrator who has been granted privileges to perform the relevant function. The TOE offers the ability to construct administrative users with a variety of different privilege level roles to aid in functional segregation. This functionality is claimed (for X.509 certificate construction) within FMT_MTD.1/CryptoKeys.

3.4.3.2 Guidance Documentation

143 The evaluator shall review the guidance documentation to determine that each of the TSF-data-manipulating functions implemented in response to the requirements of the cPP is identified, and that configuration information is provided to ensure that only administrators have access to the functions.

Findings: Section 4.1 of [SUPP] offers an introduction to the privilege roles in the TOE including any restrictive management commands that are affected by the underlying privilege role.

144 If the TOE supports handling of X.509v3 certificates and provides a trust store, the evaluator shall review the guidance documentation to determine that it provides sufficient information for the administrator to configure and maintain the trust store in a secure way. If the TOE supports loading of CA certificates, the evaluator shall review the guidance documentation to determine that it provides sufficient information for the administrator to securely load CA certificates into the trust store. The evaluator shall also review the guidance documentation to determine that it explains how to designate a CA certificate a trust anchor.

Findings: Management of the Web UI X.509 certificate is described in [SUPP] under sections 3.3.4, 3.3.5 and 3.3.6. A combination of use of the CLI and Web UI is necessary to achieve the desired effects. Trust anchors are established implicitly by ensuring that the chain ends in a self-signed root.

3.4.4 FMT_SMF.1 Specification of Management Functions

145 The security management functions for FMT_SMF.1 are distributed throughout the cPP and are included as part of the requirements in FTA_SSL_EXT.1, FTA_SSL.3, FTA_TAB.1, FMT_MOF.1/ManualUpdate, FMT_MOF.1/AutoUpdate (if included in the ST), FIA_AFL.1, FIA_X509_EXT.2.2 (if included in the ST), FPT_TUD_EXT.1.2 & FPT_TUD_EXT.2.2 (if included in the ST and if they include an administrator-configurable action), FMT_MOF.1/Services, and FMT_MOF.1/Functions (for all of these SFRs that are included in the ST), FMT_MTD, FPT_TST_EXT, and any cryptographic management functions specified in the reference standards. Compliance to these requirements satisfies compliance with FMT_SMF.1.

3.4.4.1 TSS (containing also requirements on Guidance Documentation and Tests)

146 The evaluator shall examine the TSS, Guidance Documentation and the TOE as observed during all other testing and shall confirm that the management functions specified in FMT_SMF.1 are provided by the TOE.

Findings: In the [ST] in section 5.2, the set of functions available to manage the TOE are listed. For each function, we found a corresponding description of the function described in the set of guidance documents.

a) Ability to administer the TOE locally and remotely: this is performed over the local and remote interfaces as described in [SUPP] in section 3.2 and supported by additional information from [ADMIN], [CLI] and the various hardware guides ([HW190] and [HWx95]) (see information provided in work units for FIA_UIA_EXT.1).

b) Ability to configure the access banner: The CLI and web access banners are configured as per section 4.4 of the [SUPP].

c) Ability to configure the session inactivity time before session termination or locking: Session inactivity termination settings for both Web UI and CLI (local and remote) are configured as per section 4.5.2 of [SUPP].

d) Ability to update the TOE and to verify the updates: TOE upgrades and verification is performed according to the instructions provided in section 2.1 of [SUPP]. More information can be found in work units for FMT_MOF.1/ManualUpdate.

e) Ability to configure the authentication failure parameters: Authentication failure handling is described in [SUPP] section 4.5.1. More information can be found in work units for FIA_AFL.1.

f) Ability to configure audit behaviour: Information about modifying the remote logging parameters is provided in [SUPP] section 3.4.

g) Ability to manage the cryptographic keys: The TOE can generate private keys for its own use with the web interface X.509 certificate. Instructions for generating this key is found in section 3.3.4 of [SUPP].

h) Ability to configure the cryptographic functionality: Cryptographic parameters for both SSH and TLS can be configured. SSH parameters are described in section 3.3.3 of [SUPP] and TLS parameters are described in section 3.3.3 of [SUPP].

i) Ability to re-enable an Administrator account: This function is described in section 4.5.1 of [SUPP]. Additional information can be found in work units for FIA_AFL.1.

j) Ability to set the time which is used for time-stamps: Being able to set the date/time is described in section 4.6 of [SUPP].

k) Ability to manage the TOE's trust store and designate X509.v3 certificates as trust anchors: This information is provided in [SUPP] section 3.3.5. Additional information is provided in work units for FMT_MTD.1/CoreData.

l) Ability to import X.509v3 certificates to the TOE's trust store: This information is provided in [SUPP] section 3.3.5. Additional information is provided in work units for FMT_MTD.1/CoreData.

147 The evaluator shall confirm that the TSS details which security management functions are available through which interface(s) (local administration interface, remote administration interface).

Findings:	In the [ST] section 6.1 under heading FIA_SMF.1, the TSS identifies the TOE management interfaces and the interfaces they can be used at.

148 For distributed TOEs with the option 'ability to configure the interaction between TOE components' the evaluator shall examine that the ways to configure the interaction between TOE components is detailed in the TSS and Guidance Documentation. The evaluator shall check that the TOE behaviour observed during testing of the configured SFRs is as described in the TSS and Guidance Documentation.

Findings:	The TOE is not a distributed TOE.

149 **TD408 -** The evaluator shall examine the TSS and Guidance Documentation to verify they both describe the local administrative interface. The evaluator shall ensure the Guidance Documentation includes appropriate warnings for the administrator to ensure the interface is local.

Findings:	The TSS describes the local console in section 6.1 of the [ST] under heading FIA_UIA_EXT.1. The [SUPP] describes the local console in section 1.5 as directly

	connected over a serial port (for physical devices) and the hypervisor console mechanism (for virtual devices).

3.4.4.2 Guidance Documentation

150 [ed.] See section 3.4.4.1 in this Assurance Activity Report.

3.4.4.3 Tests

151 The evaluator tests management functions as part of testing the SFRs identified in section 2.4.4. No separate testing for FMT_SMF.1 is required unless one of the management functions in FMT_SMF.1.1 has not already been exercised under any other SFR.

Findings:	All management functions have been tested within other SFRs.

3.4.5 FMT_SMR.2 Restrictions on security roles

3.4.5.1 Guidance Documentation

152 The evaluator shall review the guidance documentation to ensure that it contains instructions for administering the TOE both locally and remotely, including any configuration that needs to be performed on the client for remote administration.

Findings:	Information about how to log onto the TOE over the remote SSH CLI can be found in [CLI] in section "Accessing the Command Line Interface (CLI)". This information pertains to both physical and virtual images. While that section references use of telnet, the [SUPP] in section 3.3 informs the end-user not to use telnet. Logging onto the device using the remote Web UI interface is provided in [ADMIN] in chapter 2 ("Accessing the Appliance"). This information is applicable to both physical and virtual devices. Local console access for physical appliances described in [HW170] can be found in the section titled "Connecting the Interface Cables and Verifying Connectivity". In the [HWx95] document, this information can be found in the section titled "Log In to the Appliance Using the CLI". Local console requirements about baud rate, etc. are described. The local console use for virtual appliances for ESXi is briefly described in [VAPP] in section "Deploy on VMWare ESXi". Further information about the ESXi hypervisor and how to access the console of a virtual device would be available from VMWare.

3.4.5.2 Tests

153 In the course of performing the testing activities for the evaluation, the evaluator shall use all supported interfaces, although it is not necessary to repeat each test involving an administrative action with each interface. The evaluator shall ensure, however, that each supported method of administering the TOE that conforms to the requirements of this cPP be tested; for instance, if the TOE can be administered through a local hardware interface; SSH; and TLS/HTTPS; then all three methods of administration must be exercised during the evaluation team's test activities.

Findings:	All interfaces are tested as described in the guidance documentation.

3.5 Protection of the TSF (FPT)

3.5.1 FPT_SKP_EXT.1 Protection of TSF Data (for reading of all pre-shared, symmetric and private keys)

3.5.1.1 TSS

154 The evaluator shall examine the TSS to determine that it details how any pre-shared keys, symmetric keys, and private keys are stored and that they are unable to be viewed through an interface designed specifically for that purpose, as outlined in the application note. If these values are not stored in plaintext, the TSS shall describe how they are protected/obscured.

Findings:	This information is found in the [ST] section 6.1 under heading FPT_APW_EXT.1 and FPT_SKP_EXT.1. When used in the evaluated configuration, there are no facilities to view plaintext passwords or keys. Use of specific CLI commands are necessary at setup time to ensure that such keys and passwords are stored using cryptographic means in the various configuration files.

3.5.2 FPT_APW_EXT.1 Protection of Administrator Passwords

3.5.2.1 TSS

155 The evaluator shall examine the TSS to determine that it details all authentication data that are subject to this requirement, and the method used to obscure the plaintext password data when stored. The TSS shall also detail passwords are stored in such a way that they are unable to be viewed through an interface designed specifically for that purpose, as outlined in the application note.

Findings:	This information is found in the [ST] section 6.1 under heading FPT_APW_EXT.1 and FPT_SKP_EXT.1. When used in the evaluated configuration, there are no facilities to view plaintext passwords or keys. Use of specific CLI commands are necessary at setup time to ensure that such keys and passwords are stored using cryptographic means in the various configuration files.

3.5.3 FPT_TST_EXT.1 TSF testing

3.5.3.1 TSS

156 The evaluator shall examine the TSS to ensure that it details the self-tests that are run by the TSF; this description should include an outline of what the tests are actually doing (e.g., rather than saying "memory is tested", a description similar to "memory is tested by writing a value to each memory location and reading it back to ensure it is identical to what was written" shall be used). The evaluator shall ensure that the TSS makes an argument that the tests are sufficient to demonstrate that the TSF is operating correctly.

Findings:	The [ST] section 6.1 under heading FPT_TST_EXT.1. The TSS makes a compelling argument that the tests are sufficient to demonstrate that the TSF is operating correctly.

157 For distributed TOEs the evaluator shall examine the TSS to ensure that it details which TOE component performs which self-tests and when these self-tests are run.

Findings:	The TOE is not a distributed TOE.

3.5.3.2 Guidance Documentation

158 The evaluator shall also ensure that the guidance documentation describes the possible errors that may result from such tests, and actions the administrator should take in response; these possible errors shall correspond to those described in the TSS.

Findings:	The [SUPP] in section 7 offers a summary of the modes of operation of the TOE. Specific errors related to power-on self-tests are described in section 3.3.7 of [SUPP] as well as the steps needed to respond to such errors.

159 For distributed TOEs the evaluator shall ensure that the guidance documentation describes how to determine from an error message returned which TOE component has failed the self-test.

Findings:	The TOE is not a distributed TOE.

3.5.3.3 Tests

160 It is expected that at least the following tests are performed:

a. Verification of the integrity of the firmware and executable software of the TOE

b. Verification of the correct operation of the cryptographic functions necessary to fulfil any of the SFRs.

161 Although formal compliance is not mandated, the self-tests performed should aim for a level of confidence comparable to:

a. [FIPS 140-2], chap. 4.9.1, Software/firmware integrity test for the verification of the integrity of the firmware and executable software. Note that the testing is not restricted to the cryptographic functions of the TOE.

b. [FIPS 140-2], chap. 4.9.1, Cryptographic algorithm test for the verification of the correct operation of cryptographic functions. Alternatively, national requirements of any CCRA member state for the security evaluation of cryptographic functions should be considered as appropriate.

162 The evaluator shall either verify that the self-tests described above are carried out during initial start-up or that the developer has justified any deviation from this.

High-Level Test Description
The vendor provided help on testing that the secure boot functionality could be tested by purposely invoking an error state.
PASS

163 For distributed TOEs the evaluator shall perform testing of self-tests on all TOE components according to the description in the TSS about which self-test are performed by which component.

Findings:	The TOE is not a distributed TOE.

3.5.4 FPT_TUD_EXT.1 Trusted Update

3.5.4.1 TSS

164 The evaluator shall verify that the TSS describe how to query the currently active version. If a trusted update can be installed on the TOE with a delayed activation, the TSS needs to describe how and when the inactive version becomes active. The evaluator shall verify this description.

Findings:	Section 6.1 of the [ST] under FPT_TUD_EXT.1 describes that both the CLI and GUI can be used to query the currently active version of the TOE software/firmware. Updates to the TOE are not delayed.

165 The evaluator shall verify that the TSS describes all TSF software update mechanisms for updating the system firmware and software (for simplicity the term 'software' will be used in the following although the requirements apply to firmware and software). The evaluator shall verify that the description includes a digital signature verification of the software before installation and that installation fails if the verification fails. Alternatively an approach using a published hash can be used. In this case the TSS shall detail this mechanism instead of the digital signature verification mechanism. The evaluator shall verify that the TSS describes the method by which the digital signature or published hash is verified to include how the candidate updates are obtained, the processing associated with verifying the digital signature or published hash of the update, and the actions that take place for both successful and unsuccessful signature verification or published hash verification.

Findings:	Section 6.1 of the [ST] under FPT_TUD_EXT.1 describes that the TOE uses a published SHA2-384 hash to validate the TOE firmware. The administrator is required to manually verify the hash before confirming that the new image can be installed.

166 If the options ‘support automatic checking for updates’ or ‘support automatic updates’ are chosen from the selection in FPT_TUD_EXT.1.2, the evaluator shall verify that the TSS explains what actions are involved in automatic checking or automatic updating by the TOE, respectively.

Findings:	Section 5.2 of the [ST] has not selected this option for FPT_TUD_EXT.1.2.

167 For distributed TOEs, the evaluator shall examine the TSS to ensure that it describes how all TOE components are updated, that it describes all mechanisms that support continuous proper functioning of the TOE during update (when applying updates separately to individual TOE components) and how verification of the signature or checksum is performed for each TOE component. Alternatively, this description can be provided in the guidance documentation. In that case the evaluator should examine the guidance documentation instead.

Findings:	The TOE is not a distributed TOE.

168 If the ST author indicates that a certificate-based mechanism is used for software update digital signature verification, the evaluator shall verify that the TSS contains a description of how the certificates are contained on the device. The evaluator also ensures that the TSS (or guidance documentation) describes how the certificates are installed/updated/selected, if necessary.

Findings:	Certificate-based verification mechanisms are not claimed by the [ST].

169 If a published hash is used to protect the trusted update mechanism, then the evaluator shall verify that the trusted update mechanism does involve an active authorization step of the Security Administrator, and that download of the published hash value, hash comparison and update is not a fully automated process involving no active authorization by the Security Administrator. In particular, authentication as Security Administration according to FMT_MOF.1/ManualUpdate needs to be part of the update process when using published hashes.

Findings: Section 6.1 of the [ST] in section FPT_TUD_EXT.1 indicates that an administrator is required to actively confirm manual installation of the new image after an independent check has been conducted.

3.5.4.2 Guidance Documentation

170 The evaluator shall verify that the guidance documentation describes how to query the currently active version. If a trusted update can be installed on the TOE with a delayed activation, the guidance documentation needs to describe how to query the loaded but inactive version.

Findings: Section 2.1 of the [SUPP] directs administrators to use the CLI “version” command to check the version. The version can also be checked using the GUI as described in [ADMIN] section “Displaying the Version Information for AsyncOS”. The TOE does not install updates using a delayed activation.

171 The evaluator shall verify that the guidance documentation describes how the verification of the authenticity of the update is performed (digital signature verification or verification of published hash). The description shall include the procedures for successful and unsuccessful verification. The description shall correspond to the description in the TSS.

Findings: This information is provided in [SUPP] section 2.1. When performing an update using the online Cisco update servers, the procedure to validate the hash requires a bit more involvement from the administrator. This information is provided in [SUPP] section 2.1.

172 If a published hash is used to protect the trusted update mechanism, the evaluator shall verify that the guidance documentation describes how the Security Administrator can obtain authentic published hash values for the updates.

Findings: According to section 2.1 of [SUPP], the authentic hash is provided by the Cisco update servers in the manifest file. This information can be viewed in the upgrade_logs and updater_logs. This information is retrieved over HTTPS from the Cisco upgrade servers.

173 For distributed TOEs the evaluator shall verify that the guidance documentation describes how the versions of individual TOE components are determined for FPT_TUD_EXT.1, how all TOE components are updated, and the error conditions that may arise from checking or applying the update (e.g. failure of signature verification, or exceeding available storage space) along with appropriate recovery actions. . The guidance documentation only has to describe the procedures relevant for the user; it does not need to give information about the internal communication that takes place when applying updates.

Findings: The TOE is not a distributed TOE.

174 If this was information was not provided in the TSS: For distributed TOEs, the evaluator shall examine the Guidance Documentation to ensure that it describes how all TOE components are updated, that it describes all mechanisms that support

continuous proper functioning of the TOE during update (when applying updates separately to individual TOE components) and how verification of the signature or checksum is performed for each TOE component.

Findings:	The TOE is not a distributed TOE.

175 If this was information was not provided in the TSS: If the ST author indicates that a certificate-based mechanism is used for software update digital signature verification, the evaluator shall verify that the Guidance Documentation contains a description of how the certificates are contained on the device. The evaluator also ensures that the Guidance Documentation describes how the certificates are installed/updated/selected, if necessary.

Findings:	The TOE does not use certificate-based mechanisms.

3.5.4.3 Tests

176 The evaluator shall perform the following tests:

a. Test 1: The evaluator performs the version verification activity to determine the current version of the product. If a trusted update can be installed on the TOE with a delayed activation, the evaluator shall also query the most recently installed version (for this test the TOE shall be in a state where these two versions match). The evaluator obtains a legitimate update using procedures described in the guidance documentation and verifies that it is successfully installed on the TOE. For some TOEs loading the update onto the TOE and activation of the update are separate steps (‘activation’ could be performed e.g. by a distinct activation step or by rebooting the device). In that case the evaluator verifies after loading the update onto the TOE but before activation of the update that the current version of the product did not change but the most recently installed version has changed to the new product version. After the update, the evaluator performs the version verification activity again to verify the version correctly corresponds to that of the update and that current version of the product and most recently installed version match again.

High-Level Test Description
Get the current version of the TOE. Attempt to install a legitimate version of the TOE for the following circumstances: an upgrade. After the install, get the current version of the TOE and ensure it is consistent with the newly installed version.
PASS

b. **TD0477 -** Test 2 [conditional]: If the TOE itself verifies a digital signature to authorize the installation of an image to update the TOE the following test shall be performed (otherwise the test shall be omitted). The evaluator first confirms that no updates are pending and then performs the version verification activity to determine the current version of the product, verifying that it is different from the version claimed in the update(s) to be used in this test. The evaluator obtains or produces illegitimate updates as defined below, and attempts to install them on the TOE. The evaluator verifies that the TOE rejects all of the illegitimate updates. The evaluator performs this test using all of the following forms of illegitimate updates:

1) A modified version (e.g. using a hex editor) of a legitimately signed update

2) An image that has not been signed

3) An image signed with an invalid signature (e.g. by using a different key as expected for creating the signature or by manual modification of a legitimate signature)

4) If the TOE allows a delayed activation of updates the TOE must be able to display both the currently executing version and most recently installed version. The handling of version information of the most recently installed version might differ between different TOEs depending on the point in time when an attempted update is rejected. The evaluator shall verify that the TOE handles the most recently installed version information for that case as described in the guidance documentation. After the TOE has rejected the update the evaluator shall verify, that both, current version and most recently installed version, reflect the same version information as prior to the update attempt.

Findings:	The TOE claimed published hash.

c. **TD0477 -** Test 3 [conditional]: If the TOE itself verifies a hash value over an image against a published hash value (i.e. reference value) that has been imported to the TOE from outside such that the TOE itself authorizes the installation of an image to update the TOE, the following test shall be performed (otherwise the test shall be omitted). If the published hash is provided to the TOE by the Security Administrator and the verification of the hash value over the update file(s) against the published hash is performed by the TOE, then the evaluator shall perform the following tests. The evaluator first confirms that no update is pending and then performs the version verification activity to determine the current version of the product, verifying that it is different from the version claimed in the update(s) to be used in this test.

1) The evaluator obtains or produces an illegitimate update such that the hash of the update does not match the published hash. The evaluator provides the published hash value to the TOE and calculates the hash of the update either on the TOE itself (if that functionality is provided by the TOE), or else outside the TOE. The evaluator confirms that the hash values are different, and attempts to install the update on the TOE, verifying that this fails because of the difference in hash values (and that the failure is logged). Depending on the implementation of the TOE, the TOE might not allow the user to even attempt updating the TOE after the verification of the hash value fails. In that case the verification that the hash comparison fails is regarded as sufficient verification of the correct behaviour of the TOE

2) **TD0477 -** The evaluator uses a legitimate update and tries to perform verification of the hash value without providing the published hash value to the TOE. The evaluator confirms that this attempt fails. Depending on the implementation of the TOE it might not be possible to attempt the verification of the hash value without providing a hash value to the TOE, e.g. if the hash value needs to be handed over to the TOE as a parameter in a command line message and the syntax check of the command prevents the execution of the command without providing a hash value. In that case the mechanism that

prevents the execution of this check shall be tested accordingly, e.g. that the syntax check rejects the command without providing a hash value, and the rejection of the attempt is regarded as sufficient verification of the correct behaviour of the TOE in failing to verify the hash. The evaluator then attempts to install the update on the TOE (in spite of the unsuccessful hash verification) and confirms that this fails. Depending on the implementation of the TOE, the TOE might not allow to even attempt updating the TOE after the verification of the hash value fails. In that case the verification that the hash comparison fails is regarded as sufficient verification of the correct behaviour of the TOE

3) If the TOE allows delayed activation of updates, the TOE must be able to display both the currently executing version and most recently installed version. The handling of version information of the most recently installed version might differ between different TOEs. Depending on the point in time when the attempted update is rejected, the most recently installed version might or might not be updated. The evaluator shall verify that the TOE handles the most recently installed version information for that case as described in the guidance documentation. After the TOE has rejected the update the evaluator shall verify, that both, current version and most recently installed version, reflect the same version information as prior to the update attempt.

Findings:	The ST claims published hash.

177 If the verification of the hash value over the update file(s) against the published hash is not performed by the TOE, Test 3 shall be skipped.

178 The evaluator shall perform Test 1, Test 2 and Test 3 (if applicable) for all methods supported (manual updates, automatic checking for updates, automatic updates).

Findings:	The ST claims published hash.

179 For distributed TOEs the evaluator shall perform Test 1, Test 2 and Test 3 (if applicable) for all TOE components.

Findings:	The TOE is not a distributed TOE.

3.5.5 FPT_STM_EXT.1 Reliable Time Stamps

3.5.5.1 TSS

180 The evaluator shall examine the TSS to ensure that it lists each security function that makes use of time, and that it provides a description of how the time is maintained and considered reliable in the context of each of the time related functions.

Findings:	Section 6.1 in the [ST] under FPT_STM_EXT.1 provides the necessary information. The TOE uses the underlying hardware for physical devices and synchronizes virtual clock devices with the underlying hypervisor for virtual appliance TOEs.

3.5.5.2 Guidance Documentation

181 The evaluator examines the guidance documentation to ensure it instructs the administrator how to set the time. If the TOE supports the use of an NTP server, the

guidance documentation instructs how a communication path is established between the TOE and the NTP server, and any configuration of the NTP client on the TOE to support this communication.

Findings:	The administrator is able to set the time as per the instructions provided in [SUPP] section 4.6. The ST does not claim use of an NTP server.

3.5.5.3 Tests

182 The evaluator shall perform the following tests:

a. Test 1: If the TOE supports direct setting of the time by the Security Administrator then the evaluator uses the guidance documentation to set the time. The evaluator shall then use an available interface to observe that the time was set correctly.

High-Level Test Description
Using the CLI as a privileged admin, show the current date/time and then change the current date time. Show the changed date/time is accepted. Using the Web UI as a privileged admin, show the current date/time and then change the current date time. Show the changed date/time is accepted.
PASS

b. Test 2: If the TOE supports the use of an NTP server; the evaluator shall use the guidance documentation to configure the NTP client on the TOE, and set up a communication path with the NTP server. The evaluator will observe that the NTP server has set the time to what is expected. If the TOE supports multiple protocols for establishing a connection with the NTP server, the evaluator shall perform this test using each supported protocol claimed in the guidance documentation.

Test Not Applicable	The ST does not claim NTP.

183 If the audit component of the TOE consists of several parts with independent time information, then the evaluator shall verify that the time information between the different parts are either synchronized or that it is possible for all audit information to relate the time information of the different part to one base information unambiguously.

Test Not Applicable	The TOE does not support independent time information.

3.6 TOE Access (FTA)

3.6.1 FTA_SSL_EXT.1 TSF-initiated Session Locking

3.6.1.1 Guidance Documentation

184 The evaluator shall confirm that the guidance documentation states whether local administrative session locking or termination is supported and instructions for configuring the inactivity time period.

Findings:	[SUPP] section 4.5.2 indicates that the CLI can be terminated after an idle period has elapsed. The CLI command to configure this is the “adminaccessconfig > timeout” function. This function affects the timeouts for all instances of the CLI (local and remote) and the remote Web UI.

3.6.1.2 Tests

185 The evaluator shall perform the following test:

a. Test 1: The evaluator follows the guidance documentation to configure several different values for the inactivity time period referenced in the component. For each period configured, the evaluator establishes a local interactive session with the TOE. The evaluator then observes that the session is either locked or terminated after the configured time period. If locking was selected from the component, the evaluator then ensures that re-authentication is needed when trying to unlock the session.

High-Level Test Description
Change the idle timeout to 5 minutes. Log into the device over the serial interface and SSH CLI interface. Wait for the full duration of the timeout without sending any keepalives. The session will terminate at exactly the configured time. Repeat the above steps for an 8 minute timer.
PASS

3.6.2 FTA_SSL.3 TSF-initiated Termination

3.6.2.1 Guidance Documentation

186 **TD0425 -** The evaluator shall confirm that the guidance documentation includes instructions for configuring the inactivity time period for remote administrative session termination.

Findings:	[SUPP] section 4.5.2 indicates that the CLI can be terminated after an idle period has elapsed. The CLI command to configure this is the “adminaccessconfig > timeout” function. This function affects the timeouts for all instances of the CLI (local and remote) and the remote Web UI.

3.6.2.2 Tests

187 For each method of remote administration, the evaluator shall perform the following test:

a. Test 1: The evaluator follows the guidance documentation to configure several different values for the inactivity time period referenced in the component. For each period configured, the evaluator establishes a remote interactive session with the TOE. The evaluator then observes that the session is terminated after the configured time period.

High-Level Test Description
Change the idle timeout to 5 minutes. Log into the device over the Web UI. With 30 seconds before the timeout expires, verify the session is still alive by sending a keep alive as described above in the TSFI commands. This should reset the timeout clock. The purpose is to ensure the timeout is not premature. Wait another minute. Verify the session is still alive by sending a keep alive. This should reset the timeout clock. The purpose is to ensure the timeout has been reset by the initial keep alive action above. Wait for the full duration of the timeout without sending any keepalives. The session should terminate. Repeat for a timeout value of 11 minutes. The SSH interface was tested in the previous test case since the CLI timers affect both the serial and SSH CLIs.
PASS

3.6.3 FTA_SSL.4 User-initiated Termination

3.6.3.1 Guidance Documentation

188 The evaluator shall confirm that the guidance documentation states how to terminate a local or remote interactive session.

Findings:	This information can be found in section 4.5.3 of [SUPP] and provides instructions for administrators to terminate their own sessions on both the CLI and Web UI interfaces.

3.6.3.2 Tests

189 For each method of remote administration, the evaluator shall perform the following tests:

a. Test 1: The evaluator initiates an interactive local session with the TOE. The evaluator then follows the guidance documentation to exit or log off the session and observes that the session has been terminated.

High-Level Test Description
Log into the serial console Log out using the TSFI previous discussed.

High-Level Test Description
Verify that the session has been terminated.
PASS

b. Test 2: The evaluator initiates an interactive remote session with the TOE. The evaluator then follows the guidance documentation to exit or log off the session and observes that the session has been terminated.

High-Level Test Description
Log into the SSH interface and logout and show that the TOE is no longer connected. Log into the Web GUI interface. Copy the URL presented. Log out using the TSFI previous discussed. Paste the URL back into the web browser and attempt to navigate directly to it.
PASS

3.6.4 FTA_TAB.1 Default TOE Access Banners

3.6.4.1 TSS

190 The evaluator shall check the TSS to ensure that it details each administrative method of access (local and remote) available to the Security Administrator (e.g., serial port, SSH, HTTPS). The evaluator shall check the TSS to ensure that all administrative methods of access available to the Security Administrator are listed and that the TSS states that the TOE is displaying an advisory notice and a consent warning message for each administrative method of access. The advisory notice and the consent warning message might be different for different administrative methods of access, and might be configured during initial configuration (e.g. via configuration file).

Findings:	This information is provided in the [ST] in section 6.1 under FTA_TAB.1. The Authorized Administrator defines a custom login banner that will be displayed at the GUI and the CLI for both local and remote access configurations prior to allowing Authorized Administrator access through those interfaces.

3.6.4.2 Guidance Documentation

191 The evaluator shall check the guidance documentation to ensure that it describes how to configure the banner message.

Findings:	The [SUPP] in section 4.4 provides the CLI command needed to set the TOE banner. This banner affects all interfaces simultaneously.

3.6.4.3 Tests

192 The evaluator shall also perform the following test:

a. Test 1: The evaluator follows the guidance documentation to configure a notice and consent warning message. The evaluator shall then, for each method of access specified in the TSS, establish a session with the TOE. The evaluator shall verify that the notice and consent warning message is displayed in each instance.

High-Level Test Description
Log into the CLI interface. Change the banner to a random string. Log into fresh sessions for all interactive interfaces and show that the banner was modified and is presented prior to I&A.
PASS

3.7 Trusted path/channels (FTP)

3.7.1 FTP_ITC.1 Inter-TSF trusted channel

3.7.1.1 TSS

193 The evaluator shall examine the TSS to determine that, for all communications with authorized IT entities identified in the requirement, each secure communication mechanism is identified in terms of the allowed protocols for that IT entity, whether the TOE acts as a server or a client, and the method of assured identification of the non-TSF endpoint. The evaluator shall also confirm that all secure communication mechanisms are described in sufficient detail to allow the evaluator to match them to the cryptographic protocol Security Functional Requirements listed in the ST.

Findings:	The TOE acts as a client for providing logging messages over scp as described in section 6.1 of the [ST] under FPT_ITC.1. In section 6.1 of the [ST] under FCS_SSHC_EXT.1, the TSS describes that the scp connection to the remote entity is assured via the use of SSHv2 host keys (which employ public/private key cryptography). As there is only one trusted channel, it is trivial to map this function to the use of FCS_SSHC_EXT.1 SFR as described in section 6.1 of the [ST] under FCS_SSHC_EXT.1

3.7.1.2 Guidance Documentation

194 The evaluator shall confirm that the guidance documentation contains instructions for establishing the allowed protocols with each authorized IT entity, and that it contains recovery instructions should a connection be unintentionally broken.

Findings:	The ability for the administrator to configure remote logging over scp is described in [SUPP] section 3.4. If the SSH connection to the SCP server on a remote syslog server fails, the log files will remain on the TOE until the connection is restored. On the next SCP Push based on either the maximum log file size being exceeded or on the time interval, the current log file and the log files previously unsuccessfully transferred will be transferred.

3.7.1.3 Tests

195 The vendor shall provide to the evaluator application layer configuration settings for all secure communication mechanisms specified by the FTP_ITC.1 requirement. This information should be sufficiently detailed to allow the evaluator to determine the application layer timeout settings for each cryptographic protocol. There is no expectation that this information must be recorded in any public-facing document or report.

196 The evaluator shall perform the following tests:

a. Test 1: The evaluators shall ensure that communications using each protocol with each authorized IT entity is tested during the course of the evaluation, setting up the connections as described in the guidance documentation and ensuring that communication is successful.

Note	The TOE maintains trusted channels to the remote audit log which is set up as per the evaluated configuration. It is constantly tested throughout the evaluation.

b. Test 2: For each protocol that the TOE can initiate as defined in the requirement, the evaluator shall follow the guidance documentation to ensure that in fact the communication channel can be initiated from the TOE.

High-Level Test Description
Engage wireshark over the appropriate interface. Within the TOE, execute a log rollover on a log configured to transmit to the device we are watching with wireshark. Examine wireshark and verify that the TOE initiates SSH communications with the logging endpoint. Examine wireshark and verify that the traffic is encrypted.
PASS

c. Test 3: The evaluator shall ensure, for each communication channel with an authorized IT entity, the channel data is not sent in plaintext.

Note	See previous test case.

d. Test 4: Objective: The objective of this test is to ensure that the TOE reacts appropriately to any connection outage or interruption of the route to the external IT entities.

The evaluator shall, for each instance where the TOE acts as a client utilizing a secure communication mechanism with a distinct IT entity, physically interrupt the connection of that IT entity for the following durations: i) a duration that exceeds the TOE's application layer timeout setting, ii) a duration shorter than the application layer timeout but of sufficient length to interrupt the MAC layer.

The evaluator shall ensure that, when the physical connectivity is restored, communications are appropriately protected and no TSF data is sent in plaintext.

In the case where the TOE is able to detect when the cable is removed from the device, another physical network device (e.g. a core switch) shall be used to interrupt the connection between the TOE and the distinct IT entity. The interruption shall not be performed at the virtual node (e.g. virtual switch) and must be physical in nature.

High-Level Test Description
Configure the TOE to send logs once a minute. Indirectly disconnect the TOE from the switch and wait for the TOE to try to transmit logs to the disconnected endpoint. The TOE should report an error. Slow down the network using a bandwidth limiter and when the TOE starts a log rollover, (indirectly) disconnect the TOE from the switch and immediately reconnect. The TOE should continue sending traffic with only a minor pause. Slow down the network using a bandwidth limiter and when the TOE starts a log rollover, (indirectly) disconnect the TOE from the switch and wait until the TOE reports an error in the local logs.
PASS

Further assurance activities are associated with the specific protocols.

197 For distributed TOEs the evaluator shall perform tests on all TOE components according to the mapping of external secure channels to TOE components in the Security Target.

Findings:	The TOE is not a distributed TOE.

198 The vendor shall provide to the evaluator application layer configuration settings for all secure communication mechanisms specified by the FTP_ITC.1 requirement. This information should be sufficiently detailed to allow the evaluator to determine the application layer timeout settings for each cryptographic protocol. There is no expectation that this information must be recorded in any public- facing document or report.

3.7.2 FTP_TRP.1/Admin Trusted Path

3.7.2.1 TSS

199 The evaluator shall examine the TSS to determine that the methods of remote TOE administration are indicated, along with how those communications are protected. The evaluator shall also confirm that all protocols listed in the TSS in support of TOE administration are consistent with those specified in the requirement, and are included in the requirements in the ST.

Findings:	Section 6.1 of the [ST] under FTP_TRP.1/Admin reiterates that the TOE offers remote administrative capabilities over a remote CLI (via SSH) and over a remote Web UI (via TLS/HTTPS). The information in the TSS is consistent with the cryptographic channel claims made in section 5.2 of the [ST].

3.7.2.2 Guidance Documentation

200 The evaluator shall confirm that the guidance documentation contains instructions for establishing the remote administrative sessions for each supported method.

Findings:	Information about how to log onto the TOE over the remote SSH CLI can be found in [CLI] in section "Accessing the Command Line Interface (CLI)". This information pertains to both physical and virtual images. While that section references use of telnet, the [SUPP] in section 3.3 informs the end-user not to use telnet. Logging onto the device using the remote Web UI interface is provided in [ADMIN] in chapter 2 ("Accessing the Appliance"). This information is applicable to both physical and virtual devices.

3.7.2.3 Tests

201 The evaluator shall perform the following tests:

a. Test 1: The evaluators shall ensure that communications using each specified (in the guidance documentation) remote administration method is tested during the course of the evaluation, setting up the connections as described in the guidance documentation and ensuring that communication is successful.

Note	The only trusted paths are the SSH interface and web interface, which are both set up as per the evaluated configuration. They are constantly tested throughout the evaluation.

b. Test 2: The evaluator shall ensure, for each communication channel, the channel data is not sent in plaintext.

High-Level Test Description
Engage wireshark over the appropriate interface. Log into the trusted path. Examine wireshark and verify that the trusted path sends encrypted traffic after any initial plaintext protocol negotiation occurs.
PASS

202 Further assurance activities are associated with the specific protocols.

203 For distributed TOEs the evaluator shall perform tests on all TOE components according to the mapping of trusted paths to TOE components in the Security Target.

Findings:	The TOE is not a distributed TOE.

4 Evaluation Activities for Optional Requirements

204 No optional requirements have been claimed.

5 Evaluation Activities for Selection-Based Requirements

5.1 Cryptographic Support (FCS)

5.1.1 FCS_HTTPS_EXT.1 HTTPS Protocol

5.1.1.1 TSS

205 The evaluator shall examine the TSS and determine that enough detail is provided to explain how the implementation complies with RFC 2818.

Findings:	Section 6.1 of the [ST] under FCS_HTTPS_EXT.1 states that the TOE is conformant with RFC 2818. RFC2818 provides guidance that servers should implement an RFC-conformant version of TLS, that the server should be able to negotiate TLS (either using a distinct port or through a connection upgrade feature) and that the server should offer an appropriate certificate to ensure clients can confirm the identity. The TSS information in the [ST] section 6.1 clearly indicates that a TLS connection is used (which is reliant on claiming FCS_TLSS_EXT.1 and requires an RFC-conformant TLS implementation). The TSS in section 6.1 of the [ST] also provides that the server has an X.509 certificate to offer to remote clients. The use of a distinct port is not a requirement of RFC 2818, though, we have confirmed through testing that the standard port 80 is closed and port 443 is active and only accepts TLS binary protocols.

5.1.1.2 Tests

206 The evaluator shall perform the following tests:

c. Test 1: The evaluator shall attempt to establish each trusted path or channel that utilizes HTTPS, observe the traffic with a packet analyser, verify that the connection succeeds, and verify that the traffic is identified as TLS or HTTPS.

Findings:	This test was conducted as part of FPT_TRP.1/Admin.

207 Other tests are performed in conjunction with the TLS evaluation activities.

208 If the TOE is an HTTPS client or an HTTPS server utilizing X.509 client authentication, then the certificate validity shall be tested in accordance with testing performed for FIA_X509_EXT.1.

5.1.2 FCS_SSHC_EXT.1 SSH Client

5.1.2.1 TSS

FCS_SSHC_EXT.1.2

209 The evaluator shall check to ensure that the TSS contains a description of the public key algorithms that are acceptable for use for authentication and that this list

conforms to FCS_SSHC_EXT.1.5. and ensure that if password-based authentication methods have been selected in the ST then these are also described.

Findings:	Section 6.1 of the [ST] under FCS_SSHC_EXT.1 identifies the public key algorithms which are suitable for authenticating. The TOE does not offer password-based authentication to remote servers. The public key algorithms claimed in FCS_SSHC_EXT.1.5 are the *hostkey* algorithms which can differ from the algorithms offered by the TOE client to the remote server when used to identify and authenticate the client to the remote server. (For the purposes of FCS_SSHC_EXT.1 and TD0411 and TD0412, this AA is assumed to refer to the use of hostkey algorithms and not those offered for use by the TOE for identifying and authenticating to a remote server.) With that said, the [ST] in section 6.1 under FCS_SSHC_EXT.1 indicates that the TOE client will only negotiate ssh-rsa hostkeys and can offer ssh-rsa, ecdh-sha2-nistp256, , ecdh-sha2-nistp384 and ecdh-sha2-nistp512 public keys for authentication to the remote server.

FCS_SSHC_EXT.1.3

210 The evaluator shall check that the TSS describes how “large packets” in terms of RFC 4253 are detected and handled.

Findings:	This information is provided in section 6.1 of the [ST] under FCS_SSHC_EXT.1. The TOE drops packets larger than 256KB.

FCS_SSHC_EXT.1.4

211 The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that optional characteristics are specified, and the encryption algorithms supported are specified as well. The evaluator shall check the TSS to ensure that the encryption algorithms specified are identical to those listed for this component.

Findings:	No optional characteristics are defined. Section 6.1 of the [ST] under FCS_SSHC_EXT.1 states that the TOE utilises aes-ctr-128, aes-ctr-256, aes-cbc-128 and aes-cbc-256 for SSH encryption. These are identical to the claims made in the SFR in section 5.2 of the [ST].

FCS_SSHC_EXT.1.5

212 The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that optional characteristics are specified, and the public key algorithms supported are specified as well. The evaluator shall check the TSS to ensure that the public key algorithms specified are identical to those listed for this component.

Findings:	No optional characteristics are defined. Section 6.1 of the [ST] under FCS_SSHC_EXT.1 states that the TOE utilises ssh-rsa for hostkey public key authentication of the remote server. This is identical to the claims made in the SFR in section 5.2 of the [ST].

FCS_SSHC_EXT.1.6

213 The evaluator shall check the TSS to ensure that it lists the supported data integrity algorithms, and that that list corresponds to the list in this component.

Findings:	The integrity algorithms are described in section 6.1 of the [ST] under FCS_SSHC_EXT.1 as HMAC-SHA. This is consistent with the SFR in section 5.2.

FCS_SSHC_EXT.1.7

214 The evaluator shall check the TSS to ensure that it lists the supported key exchange algorithms, and that that list corresponds to the list in this component.

Findings:	The key exchange algorithms are described in section 6.1 of the [ST] under FCS_SSHC_EXT.1 as diffie-hellman-group14-sha1, diffie-hellman-group16-sha512, ecdh-sha2-nistp256, ecdh-sha2-nistp384, ecdh-sha2-nistp521. This is consistent with the SFR in section 5.2.

FCS_SSHC_EXT.1.8

215 The evaluator shall check that the TSS specifies the following:

a) Both thresholds are checked by the TOE.

b) Rekeying is performed upon reaching the threshold that is hit first.

Findings:	The [ST] in section 6.1 under FCS_SSHC_EXT.1 claims "[a] rekey occurs after a threshold of no longer than one hour and no more than one gigabyte of transmitted data."

5.1.2.2 Guidance Documentation

FCS_SSHC_EXT.1.4

216 The evaluator shall also check the guidance documentation to ensure that it contains instructions on configuring the TOE so that SSH conforms to the description in the TSS (for instance, the set of algorithms advertised by the TOE may have to be restricted to meet the requirements).

Findings:	Section 3.3.2 of the [SUPP] offers the administrator guidance to select the algorithms used for the SSH client. This includes configurable ciphers.

FCS_SSHC_EXT.1.5

217 The evaluator shall also check the guidance documentation to ensure that it contains instructions on configuring the TOE so that SSH conforms to the description in the TSS (for instance, the set of algorithms advertised by the TOE may have to be restricted to meet the requirements).

Findings:	The set of hostkey public key algorithms which can be negotiated by the TOE SSH client are not configurable as per the statement in section 3.2.2 of the [SUPP] which states that the TOE SSH client will always use ssh-rsa.

FCS_SSHC_EXT.1.6

218 The evaluator shall also check the guidance documentation to ensure that it contains instructions to the administrator on how to ensure that only the allowed data integrity algorithms are used in SSH connections with the TOE (specifically, that the "none" MAC algorithm is not allowed).

Findings:	Section 3.3.2 of the [SUPP] offers the administrator guidance to select the algorithms used for the SSH client. This includes configurable integrity algorithms. The "none" MAC is not an available selection.

FCS_SSHC_EXT.1.7

219 The evaluator shall also check the guidance documentation to ensure that it contains instructions to the administrator on how to ensure that only the allowed key exchange algorithms are used in SSH connections with the TOE.

Findings:	Section 3.3.2 of the [SUPP] offers the administrator guidance to select the algorithms used for the SSH client. This includes configurable key exchange algorithms.

FCS_SSHC_EXT.1.8

220 If one or more thresholds that are checked by the TOE to fulfil the SFR are configurable, then the evaluator shall check that the guidance documentation describes how to configure those thresholds. Either the allowed values are specified in the guidance documentation and must not exceed the limits specified in the SFR (one hour of session time, one gigabyte of transmitted traffic) or the TOE must not accept values beyond the limits specified in the SFR. The evaluator shall check that the guidance documentation describes that the TOE reacts to the first threshold reached.

Findings:	These thresholds are not configurable within the TOE.

5.1.2.3 Tests

FCS_SSHC_EXT.1.2

221 Test 1: If password-based authentication methods have been selected in the ST then using the guidance documentation, the evaluator shall configure the TOE to perform password-based authentication to an SSH server, and demonstrate that a user can be successfully authenticated by the TOE to an SSH server using a password as an authenticator.

Note: Public key authentication is tested as part of testing for FCS_SSHC_EXT.1.5

Findings:	The ST only claims public key mechanisms.

FCS_SSHC_EXT.1.3

222 The evaluator shall demonstrate that if the TOE receives a packet larger than that specified in this component, that packet is dropped.

High-Level Test Description
Using a custom server tool, force the TOE to receive a packet larger than the expected TOE buffer size and show that the TOE rejects the packet in some way.
PASS

FCS_SSHC_EXT.1.4

223 The evaluator must ensure that only claimed ciphers and cryptographic primitives are used to establish a SSH connection. To verify this, the evaluator shall start session establishment for a SSH connection with a remote server (referred to as ‘remote endpoint’ below). The evaluator shall capture the traffic exchanged between the TOE and the remote endpoint during protocol negotiation (e.g. using a packet capture tool or information provided by the endpoint, respectively). The evaluator shall verify from the captured traffic that the TOE offers all the ciphers defined in the TSS for the TOE for SSH sessions, but no additional ones compared to the definition in the TSS. The evaluator shall perform one successful negotiation of an SSH session to verify that the TOE behaves as expected. It is sufficient to observe the successful negotiation of the session to satisfy the intent of the test. If the evaluator detects that not all ciphers defined in the TSS for SSH are supported by the TOE and/or the TOE supports one or more additional ciphers not defined in the TSS for SSH, the test shall be regarded as failed.

High-Level Test Description
Permit the TOE client to connect to a test SSH server and capture the TOE client's advertised supported cipher algorithms. Verify that the advertised set matches the claimed set. Forcibly use an SSH server to permit connections from the TOE client using only one of those claimed ciphers and show that the connection is successful.
PASS

FCS_SSHC_EXT.1.5

224 Test 1: The evaluator shall establish a SSH connection using each of the public key algorithms specified by the requirement to authenticate an SSH server to the TOE. It is sufficient to observe (on the wire) the successful negotiation of the algorithm to satisfy the intent of the test.

225 **TD0411 -** Test objective: The purpose of this positive test is to check the authentication of the server by the client (when establishing the transport layer connection), and not for checking generation of the authentication message from the client (in the User Authentication Protocol). The evaluator is therefore intended to establish sufficient separate SSH connections (with an appropriately configured server) to cause the TOE to demonstrate use of all public key algorithms claimed in FCS_SSHC_EXT.1.5 in the ST.

High-Level Test Description
Use the TOE client and connect to a test SSH server which only permits the specified public key algorithms in turn. This requires the TOE to be loaded with a private key corresponding to the key pair and the non-TOE server to have access to the public key half.
PASS

226 Test 2: The evaluator shall configure an SSH server to only allow a public key algorithm that is not included in the ST selection. The evaluator shall attempt to establish an SSH connection from the TOE to the SSH server and observe that the connection is rejected.

High-Level Test Description
Ensure the TOE has a supported host key for the TOE. Off-TOE, use a different private key (generated with an unsupported algorithm). Permit the TOE client to connect to the non-TOE server. The connection attempt should fail.
PASS

FCS_SSHC_EXT.1.6

227 Test 1: (conditional, if an HMAC or AEAD_AES_*_GCM algorithm is selected in the ST) The evaluator shall establish an SSH connection using each of the algorithms, except "implicit", specified by the requirement. It is sufficient to observe (on the wire) the successful negotiation of the algorithm to satisfy the intent of the test.

228 Note: To ensure the observed algorithm is used, the evaluator shall ensure a non-aes*-gcm@openssh.com encryption algorithm is negotiated while performing this test.

High-Level Test Description
Using an SSH Server, forcibly permit only the claimed integrity algorithms and show that connections by the TOE SSH client are accepted to form a successful connection.
PASS

229 Test 2: (conditional, if an HMAC or AEAD_AES_*_GCM algorithm is selected in the ST) The evaluator shall configure an SSH server to only allow a MAC algorithm that is not included in the ST selection. The evaluator shall attempt to connect from the TOE to the SSH server and observe that the attempt fails.

230 Note: To ensure the proposed MAC algorithm is used, the evaluator shall ensure a non-aes*-gcm@openssh.com encryption algorithm is negotiated while performing this test .

High-Level Test Description
Using an SSH Server, forcibly permit an integrity algorithm which is not claimed by the TOE and show that a TOE SSH client connection results in a failed connection.
PASS

FCS_SSHC_EXT.1.7

231 Test 1: The evaluator shall configure an SSH server to permit all allowed key exchange methods. The evaluator shall attempt to connect from the TOE to the SSH server using each allowed key exchange method, and observe that each attempt succeeds.

High-Level Test Description
Using an SSH server, forcibly permit only one claimed key exchange mechanism at a time and show that the TOE client will successfully connect using that algorithm.
PASS

FCS_SSHC_EXT.1.8

232 The evaluator needs to perform testing that rekeying is performed according to the description in the TSS. The evaluator shall test both, the time-based threshold and the traffic-based threshold.

233 For testing of the time-based threshold the evaluator shall use the TOE to connect to an SSH server and keep the session open until the threshold is reached. The evaluator shall verify that the SSH session has been active longer than the threshold value and shall verify that the TOE initiated a rekey (the method of verification shall be reported by the evaluator).

234 Testing does not necessarily have to be performed with the threshold configured at the maximum allowed value of one hour of session time but the value used for testing shall not exceed one hour. The evaluator needs to ensure that the rekeying has been initiated by the TOE and not by the SSH server the TOE is connected to.

High-Level Test Description
Using a custom SSH server, use the TOE client to connect to the server and trickle data over the channel to avoid disconnection due to idle timeout. Ensure that the TOE rekeys before 1 hour has elapsed. Ensure that the TOE is responsible for sending the rekey initiation.
PASS

235 **TD0475 -** For testing of the traffic-based threshold the evaluator shall use the TOE to connect to an SSH server, and shall transmit data to and/or receive data from the TOE within the active SSH session until the threshold for data protected by either encryption key is reached. It is acceptable if the rekey occurs before the threshold is reached (e.g. because the traffic is counted according to one of the alternatives given in the Application Note for FCS_SSHC_EXT.1.8).

236 The evaluator shall verify that more data has been transmitted within the SSH session than the threshold allows and shall verify that the TOE initiated a rekey (the method of verification shall be reported by the evaluator).

237 Testing does not necessarily have to be performed with the threshold configured at the maximum allowed value of one gigabyte of transferred traffic but the value used for testing shall not exceed one gigabyte. The evaluator needs to ensure that the rekeying has been initiated by the TOE and not by the SSH server the TOE is connected to.

High-Level Test Description
Using a custom SSH server, permit the TOE client to connect to the server. The server will send large amounts of data over the channel back to the client. Ensure that the TOE rekeys before 1 GB in the aggregate has been transmitted. Ensure that the TOE is responsible for sending the rekey initiation.
PASS

238 If one or more thresholds that are checked by the TOE to fulfil the SFR are configurable, the evaluator needs to verify that the threshold(s) can be configured as described in the guidance documentation and the evaluator needs to test that modification of the thresholds is restricted to Security Administrators (as required by FMT_MOF.1/Functions).

Findings:	The thresholds are not configurable.

239 In cases where data transfer threshold could not be reached due to hardware limitations it is acceptable to omit testing of this (SSH rekeying based on data transfer threshold) threshold if both the following conditions are met:

a) An argument is present in the TSS section describing this hardware-based limitation and

b) All hardware components that are the basis of such argument are definitively identified in the ST. For example, if specific Ethernet Controller or WiFi radio chip is the root cause of such limitation, these chips must be identified.

Findings:	The thresholds are not restricted by hardware limitations.

FCS_SSHC_EXT.1.9

240 Test 1: The evaluator shall delete all entries in the TOE's list of recognized SSH server host keys and, if selected, all entries in the TOE's list of trusted certification authorities. The evaluator shall initiate a connection from the TOE to an SSH server. The evaluator shall ensure that the TOE either rejects the connection or displays the SSH server's public key (either the key bytes themselves or a hash of the key using any allowed hash algorithm) and prompts the user to accept or deny the key before continuing the connection.

High-Level Test Description
Clear the known host key database. Using the TOE SSH client, connect to an SSH server and show that the TOE either warns the administrator that the host is unknown or that it rejects the connection attempt until after the host key has been manually added.
PASS

241 Test 2: The evaluator shall add an entry associating a host name with a public key into the TOE's local database. The evaluator shall replace, on the corresponding SSH server, the server's host key with a different host key. If 'password-based' is selected for the TOE in FCS_SSHC_EXT.1.2, the evaluator shall initiate a connection from the TOE to the SSH server using password-based authentication, shall ensure that the TOE rejects the connection, and shall ensure that the password was not transmitted to the SSH server (for example, by instrumenting the SSH server with a debugging capability to output received passwords). If 'password-based' is not selected for the TOE in FCS_SSHC_EXT.1.2, the evaluator shall initiate a connection from the TOE to the SSH server using public key-based authentication, and shall ensure that the TOE rejects the connection.

High-Level Test Description
Add a host key to the known hosts database either explicitly or implicitly depending on the mechanism for inserting the key. Generate a different host key for the non-TOE SSH server. Using the TOE SSH client, connect to the SSH server that advertises the wrong host key and show that the TOE rejects the connection. Verify that passwords (if claimed) are not transmitted; verify public key authentication fails.
PASS

5.1.3 FCS_SSHS_EXT.1 SSH Server

5.1.3.1 TSS

FCS_SSHS_EXT.1.2

242 The evaluator shall check to ensure that the TSS contains a description of the public key algorithms that are acceptable for use for authentication and that this list conforms to FCS_SSHS_EXT.1.5. and ensure that if password-based authentication methods have been selected in the ST then these are also described.

Findings:	Section 6.1 of the [ST] under FCS_SSHS_EXT.1 identifies the public key algorithms which are suitable for authenticating. The TOE also permits password-based authentication to itself. The public key algorithms claimed in FCS_SSHS_EXT.1.5

	are the user authentication public key algorithms and section 6.1 of the [ST] under FCS_SSHS_EXT.1 indicates that the TOE server will accept rsa-sha2-256 and, rsa-sha2-512 public keys for authentication from end users. This is consistent with the SFR in section 5.2.

FCS_SSHS_EXT.1.3

243 The evaluator shall check that the TSS describes how "large packets" in terms of RFC 4253 are detected and handled.

Findings:	This information is provided in section 6.1 of the [ST] under FCS_SSHS_EXT.1. The TOE drops packets larger than 256KB.

FCS_SSHS_EXT.1.4

244 The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that optional characteristics are specified, and the encryption algorithms supported are specified as well. The evaluator shall check the TSS to ensure that the encryption algorithms specified are identical to those listed for this component.

Findings:	No optional characteristics are defined. Section 6.1 of the [ST] under FCS_SSHS_EXT.1 states that the TOE utilises aes-ctr-128, aes-ctr-256, aes-cbc-128 and aes-cbc-256 for SSH encryption. These are identical to the claims made in the SFR in section 5.2 of the [ST].

FCS_SSHS_EXT.1.5

245 The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that optional characteristics are specified, and the public key algorithms supported are specified as well. The evaluator shall check the TSS to ensure that the public key algorithms specified are identical to those listed for this component.

Findings:	No optional characteristics are defined. Section 6.1 of the [ST] under FCS_SSHS_EXT.1 states that the TOE server will accept rsa-sha2-256 and, rsa-sha2-512 public keys for authentication from end users. This is identical to the claims made in the SFR in section 5.2 of the [ST].

FCS_SSHS_EXT.1.6

246 The evaluator shall check the TSS to ensure that it lists the supported data integrity algorithms, and that that list corresponds to the list in this component.

Findings:	The integrity algorithms are described in section 6.1 of the [ST] under FCS_SSHS_EXT.1 as HMAC-SHA. This is consistent with the SFR in section 5.2.

FCS_SSHS_EXT.1.7

247 The evaluator shall check the TSS to ensure that it lists the supported key exchange algorithms, and that that list corresponds to the list in this component.

Findings:	The key exchange algorithms are described in section 6.1 of the [ST] under FCS_SSHS_EXT.1 as diffie-hellman-group14-sha1, ecdh-sha2-nistp256 and ecdh-sha2-nistp521. This is consistent with the SFR in section 5.2.

FCS_SSHS_EXT.1.8

248 The evaluator shall check that the TSS specifies the following:

a) Both thresholds are checked by the TOE.

b) Rekeying is performed upon reaching the threshold that is hit first.

Findings:	The [ST] in section 6.1 under FCS_SSHS_EXT.1 claims “[t]he sessions keys for SSH sessions have a threshold of one hour, and no more than one gigabyte of transmitted data. A rekey is performed whenever either of the two thresholds is reached.”

5.1.3.2 Guidance Documentation

FCS_SSHS_EXT.1.4

249 The evaluator shall also check the guidance documentation to ensure that it contains instructions on configuring the TOE so that SSH conforms to the description in the TSS (for instance, the set of algorithms advertised by the TOE may have to be restricted to meet the requirements).

Findings:	Section 3.3.2 of the [SUPP] offers the administrator guidance to select the algorithms used for the SSH server. This includes configurable ciphers.

FCS_SSHS_EXT.1.5

250 The evaluator shall also check the guidance documentation to ensure that it contains instructions on configuring the TOE so that SSH conforms to the description in the TSS (for instance, the set of algorithms advertised by the TOE may have to be restricted to meet the requirements).

Findings:	Section 3.3.2 of the [SUPP] offers the administrator guidance to select the algorithms used for the SSH server. This includes configurable public key algorithms permitted to be used by authenticating clients.

FCS_SSHS_EXT.1.6

251 The evaluator shall also check the guidance documentation to ensure that it contains instructions to the administrator on how to ensure that only the allowed data integrity algorithms are used in SSH connections with the TOE (specifically, that the “none” MAC algorithm is not allowed).

Findings:	Section 3.3.2 of the [SUPP] offers the administrator guidance to select the algorithms used for the SSH server. This includes configurable integrity algorithms. The “none” MAC is not an available selection.

FCS_SSHS_EXT.1.7

252 The evaluator shall also check the guidance documentation to ensure that it contains instructions to the administrator on how to ensure that only the allowed key exchange algorithms are used in SSH connections with the TOE.

Findings:	Section 3.3.2 of the [SUPP] offers the administrator guidance to select the algorithms used for the SSH server. This includes configurable key exchange algorithms.

FCS_SSHS_EXT.1.8

253 If one or more thresholds that are checked by the TOE to fulfil the SFR are configurable, then the evaluator shall check that the guidance documentation describes how to configure those thresholds. Either the allowed values are specified in the guidance documentation and must not exceed the limits specified in the SFR (one hour of session time, one gigabyte of transmitted traffic) or the TOE must not accept values beyond the limits specified in the SFR. The evaluator shall check that the guidance documentation describes that the TOE reacts to the first threshold reached.

Findings:	These thresholds are not configurable within the TOE.

5.1.3.3 Tests

FCS_SSHS_EXT.1.2

254 Test 1: If password-based authentication methods have been selected in the ST then using the guidance documentation, the evaluator shall configure the TOE to accept password-based authentication, and demonstrate that user authentication succeeds when the correct password is provided by the user.

NOTE:	Please refer to tests in FIA_UIA_EXT.1

255 Test 2: If password-based authentication methods have been selected in the ST then the evaluator shall use an SSH client, enter an incorrect password to attempt to authenticate to the TOE, and demonstrate that the authentication fails.

256 Note: Public key authentication is tested as part of testing for FCS_SSHS_EXT.1.5.

NOTE:	Please refer to tests in FIA_UIA_EXT.1

FCS_SSHS_EXT.1.3

257 The evaluator shall demonstrate that if the TOE receives a packet larger than that specified in this component, that packet is dropped.

High-Level Test Description
Using a custom tool, transmit a packet larger than the expected TOE buffer size and show that the TOE rejects the packet in some way.
PASS

FCS_SSHS_EXT.1.4

258 The evaluator must ensure that only claimed ciphers and cryptographic primitives are used to establish a SSH connection. To verify this, the evaluator shall start session establishment for a SSH connection from a remote client (referred to as ‘remote endpoint’ below). The evaluator shall capture the traffic exchanged between the TOE and the remote endpoint during protocol negotiation (e.g. using a packet capture tool or information provided by the endpoint, respectively). The evaluator shall verify from the captured traffic that the TOE offers all the ciphers defined in the TSS for the TOE for SSH sessions, but no additional ones compared to the definition in the TSS. The evaluator shall perform one successful negotiation of an SSH session to verify that the TOE behaves as expected. It is sufficient to observe the successful negotiation of the session to satisfy the intent of the test. If the evaluator detects that not all ciphers defined in the TSS for SSH are supported by the TOE and/or the TOE supports one or more additional ciphers not defined in the TSS for SSH, the test shall be regarded as failed.

High-Level Test Description
Using an SSH client, connect to the TOE server and capture the TOE server’s advertised supported cipher algorithms. Verify that the advertised set matches the

High-Level Test Description
claimed set. Forcibly use a SSH client to connect using only one of those ciphers and show that the connection is successful.
PASS

FCS_SSHS_EXT.1.5

259 Test 1: The evaluator shall establish a SSH connection using each of the public key algorithms specified by the requirement to authenticate the TOE to an SSH client. It is sufficient to observe (on the wire) the successful negotiation of the algorithm to satisfy the intent of the test.

High-Level Test Description
Using an SSH client, connect to the TOE server using the specified public key algorithms in turn. This requires the TOE to be loaded with a public key corresponding to the key pair.
PASS

260 **TD412 -** Test objective: The purpose of this negative test is to verify that the server rejects authentication attempts of clients that present a public key that does not match public key(s) associated by the TOE with the identity of the client (i.e. the public keys are unknown to the server). To demonstrate correct functionality it is sufficient to determine that an SSH connection was not established after using a valid username and an unknown key of supported type.

261 Test 2: The evaluator shall choose one public key algorithm supported by the TOE. The evaluator shall generate a new key pair for that algorithm without configuring the TOE to recognize the public key for authentication. The evaluator shall use an SSH client to attempt to connect to the TOE with the new key pair and demonstrate that authentication fails.

High-Level Test Description
Load a supported public key into the TOE. Off-TOE, use a different private key (generated with the same public key algorithm) to try to connect. The connection attempt should fail.
PASS

262 Test 3: The evaluator shall configure an SSH client to only allow a public key algorithm that is not included in the ST selection. The evaluator shall attempt to establish an SSH connection from the SSH client to the TOE and observe that the connection is rejected.

High-Level Test Description
Building upon the previous test case, attempt to log into the TOE using a private key from an unsupported algorithm and show the user fails to log in.
PASS

FCS_SSHS_EXT.1.6

263 Test 1: (conditional, if an HMAC or AEAD_AES_*_GCM algorithm is selected in the ST) The evaluator shall establish an SSH connection using each of the algorithms, except "implicit", specified by the requirement. It is sufficient to observe (on the wire) the successful negotiation of the algorithm to satisfy the intent of the test.

264 Note: To ensure the observed algorithm is used, the evaluator shall ensure a non-aes*-gcm@openssh.com encryption algorithm is negotiated while performing this test.

High-Level Test Description
Using an SSH client, forcibly negotiate only the claimed integrity algorithms and show that they are accepted to form a successful connection.
PASS

265 Test 2: (conditional, if an HMAC or AEAD_AES_*_GCM algorithm is selected in the ST) The evaluator shall configure an SSH client to only allow a MAC algorithm that is not included in the ST selection. The evaluator shall attempt to connect from the SSH client to the TOE and observe that the attempt fails.

266 Note: To ensure the proposed MAC algorithm is used, the evaluator shall ensure a non-aes*-gcm@openssh.com encryption algorithm is negotiated while performing this test.

High-Level Test Description
Using an SSH client, forcibly negotiate an integrity algorithm which is not claimed by the TOE and show that it results in a failed connection.
PASS

FCS_SSHS_EXT.1.7

267 Test 1: The evaluator shall configure an SSH client to only allow the diffie-hellman-group1-sha1 key exchange. The evaluator shall attempt to connect from the SSH client to the TOE and observe that the attempt fails.

High-Level Test Description
Using an SSH client, forcibly negotiate the diffie-hellman-group1-sha1 key exchange algorithm which is not supported by the TOE and show that it results in a failed connection.
PASS

268 Test 2: For each allowed key exchange method, the evaluator shall configure an SSH client to only allow that method for key exchange, attempt to connect from the client to the TOE, and observe that the attempt succeeds.

High-Level Test Description
Using an SSH client, forcibly negotiate each of the claimed key exchange algorithms in turn and show that it results in a successful connection.
PASS

FCS_SSHS_EXT.1.8

269 The evaluator needs to perform testing that rekeying is performed according to the description in the TSS. The evaluator shall test both, the time-based threshold and the traffic-based threshold.

270 For testing of the time-based threshold the evaluator shall use an SSH client to connect to the TOE and keep the session open until the threshold is reached. The evaluator shall verify that the SSH session has been active longer than the threshold value and shall verify that the TOE initiated a rekey (the method of verification shall be reported by the evaluator).

271 Testing does not necessarily have to be performed with the threshold configured at the maximum allowed value of one hour of session time but the value used for testing shall not exceed one hour. The evaluator needs to ensure that the rekeying has been initiated by the TOE and not by the SSH client that is connected to the TOE.

High-Level Test Description
Using a custom SSH client, connect to the TOE and trickle data over the channel to avoid disconnection due to idle timeout. Ensure that the TOE rekeys before 1 hour has elapsed. Ensure that the TOE is responsible for sending the rekey initiation.
PASS

272 **TD0475 -** For testing of the traffic-based threshold the evaluator shall use the TOE to connect to an SSH client, and shall transmit data to and/or receive data from the TOE within the active SSH session until the threshold for data protected by either encryption key is reached. It is acceptable if the rekey occurs before the threshold is reached (e.g. because the traffic is counted according to one of the alternatives given in the Application Note for FCS_SSHS_EXT.1.8)

273 The evaluator shall verify that more data has been transmitted within the SSH session than the threshold allows and shall verify that the TOE initiated a rekey (the method of verification shall be reported by the evaluator).

274 Testing does not necessarily have to be performed with the threshold configured at the maximum allowed value of one gigabyte of transferred traffic but the value used for testing shall not exceed one gigabyte. The evaluator needs to ensure that the rekeying has been initiated by the TOE and not by the SSH client that is connected to the TOE.

High-Level Test Description
Using a custom SSH client, connect to the TOE and send large amounts of data over the channel. Ensure that the TOE rekeys before 1 GB in the aggregate has been transmitted. Ensure that the TOE is responsible for sending the rekey initiation.
PASS

275 If one or more thresholds that are checked by the TOE to fulfil the SFR are configurable, the evaluator needs to verify that the threshold(s) can be configured as described in the guidance documentation and the evaluator needs to test that modification of the thresholds is restricted to Security Administrators (as required by FMT_MOF.1/Functions).

Findings:	The thresholds are not configurable.

276 In cases where data transfer threshold could not be reached due to hardware limitations it is acceptable to omit testing of this (SSH rekeying based on data transfer threshold) threshold if both the following conditions are met:

c. An argument is present in the TSS section describing this hardware-based limitation and

d. All hardware components that are the basis of such argument are definitively identified in the ST. For example, if specific Ethernet Controller or WiFi radio chip is the root cause of such limitation, these chips must be identified.

Findings:	The thresholds are not restricted by hardware limitations.

5.1.4 FCS_TLSS_EXT.1 Extended: TLS Server Protocol

5.1.4.1 TSS

FCS_TLSS_EXT.1.1

277 The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that the ciphersuites supported are specified. The evaluator shall check the TSS to ensure that the ciphersuites specified are identical to those listed for this component.

Findings:	In the [ST] in section 6.1 for FCS_TLSS_EXT.1, the ciphersuites listed are identical to those listed in the SFR.

FCS_TLSS_EXT.1.2

278 The evaluator shall verify that the TSS contains a description of the denial of old SSL and TLS versions.

Findings:	Section 6.1 of the [ST] under FCS_TLSS_EXT.1 indicates that the TOE will deny connections from SSLv2, SSLv3, and TLSv1.0. This is consistent with the claims made in the SFR in section 5.2.

FCS_TLSS_EXT.1.3

279 **TD450 -** If using ECDHE or DHE ciphers, the evaluator shall verify that the TSS describes the key agreement parameters of the server Key Exchange message.

Findings:	Section 6.1 of the [ST] in FCS_TLSS_EXT.1 describes that the claimed DHE ciphersuites use the standard parameters P, Q, and G for key exchange.

5.1.4.2 Guidance Documentation

FCS_TLSS_EXT.1.1

280 The evaluator shall check the guidance documentation to ensure that it contains instructions on configuring the TOE so that TLS conforms to the description in the TSS (for instance, the set of ciphersuites advertised by the TOE may have to be restricted to meet the requirements).

Findings:	Section 3.3.3 of the [SUPP] offers the administrator guidance to select the algorithms used for the TLS server. This includes configurable ciphers. This set is consistent with the TSS claims in the [ST] section 6.1 for FCS_TLSS_EXT.1.

FCS_TLSS_EXT.1.2

281 The evaluator shall verify that any configuration necessary to meet the requirement must be contained in the AGD guidance.

Findings:	Section 3.3.3 of the [SUPP] offers the administrator guidance to select the protocol versions used for the TLS server.

FCS_TLSS_EXT.1.3

282 The evaluator shall verify that any configuration necessary to meet the requirement must be contained in the AGD guidance.

Findings:	The key size used for RSA key exchange or for DHE key agreement are not configurable in the TOE.

5.1.4.3 Tests

FCS_TLSS_EXT.1.1

283 Test 1: The evaluator shall establish a TLS connection using each of the ciphersuites specified by the requirement. This connection may be established as part of the establishment of a higher-level protocol, e.g., as part of an HTTPS session. It is sufficient to observe the successful negotiation of a ciphersuite to satisfy the intent of the test; it is not necessary to examine the characteristics of the encrypted traffic to discern the ciphersuite being used (for example, that the cryptographic algorithm is 128-bit AES and not 256-bit AES).

High-Level Test Description
Using a Lightship developed TLS client, connect to the TOE using the claimed ciphersuites.
PASS

284 Test 2: The evaluator shall send a Client Hello to the server with a list of ciphersuites that does not contain any of the ciphersuites in the server's ST and verify that the server denies the connection. Additionally, the evaluator shall send a Client Hello to the server containing only the TLS_NULL_WITH_NULL_NULL ciphersuite and verify that the server denies the connection.

High-Level Test Description
Using a Lightship developed TLS client, connect to the TOE using an unsupported ciphersuite. Then connect to the TOE using TLS_NULL_WITH_NULL_NULL.
PASS

285 Test 3: The evaluator shall use a client to send a key exchange message in the TLS connection that does not match the server-selected ciphersuite (for example, send an ECDHE key exchange while using the TLS_RSA_WITH_AES_128_CBC_SHA ciphersuite or send a RSA key exchange while using one of the ECDSA ciphersuites.) The evaluator shall verify that the TOE disconnects after receiving the key exchange message.

High-Level Test Description
Using a Lightship developed TLS client, connect to the TOE using a supported ciphersuite. The test tool will, at the appropriate time, send back a Client Key Exchange message that does not match the expected key exchange algorithm. For RSA key exchanges, the test tool will send back an ECDHE key exchange. For ECDHE and DHE key exchanges, the test tool will send back an RSA key exchange.
PASS

286 Test 4: The evaluator shall perform the following modifications to the traffic:

a. withdrawn

b. withdrawn

c. Modify a byte in the Client Finished handshake message, and verify that the server rejects the connection and does not send any application data.

High-Level Test Description
Using a Lightship developed TLS client, connect to the TOE and modify the first payload byte in the Client Finished message.
PASS

d. After generating a fatal alert by sending a Finished message from the client before the client sends a ChangeCipherSpec message, send a Client Hello with the session identifier from the previous test, and verify that the server denies the connection.

High-Level Test Description
Using a Lightship developed TLS client, connect to the TOE and capture the session ID sent back from the server. At the end of this initial handshake, reorder the ChangeCipherSpec and Finished messages so that the connection does not complete. Secondly, reconnect to the TOE and sent the previously captured session ID in the hopes that we can avoid the remainder of the handshake. Verify the TOE does not permit this.
PASS

e. (Test Intent: The intent of this test is to ensure that the server's TLS implementation immediately makes use of the key exchange and authentication algorithms to: a) Correctly encrypt (D)TLS Finished message and b) Encrypt every (D)TLS message after session keys are negotiated.)

The evaluator shall use one of the claimed ciphersuites to complete a successful handshake and observe transmission of properly encrypted application data. The evaluator shall verify that no Alert with alert level Fatal (2) messages were sent.

The evaluator shall verify that the Finished message (Content type hexadecimal 16 and handshake message type hexadecimal 14) is sent immediately after the server's ChangeCipherSpec (Content type hexadecimal 14) message. The evaluator shall examine the Finished message (encrypted example in hexadecimal of a TLS record containing a Finished message, 16 03 03 00 40 11 22 33 44 55...) and confirm that it does not contain unencrypted data (unencrypted example in hexadecimal of a TLS record containing a Finished message, 16 03 03 00 40 14 00 00 0c...), by verifying that the first byte of the encrypted Finished message does not equal hexadecimal 14 for at least one of three test messages. There is a chance that an encrypted Finished message contains a hexadecimal value of '14' at the position where a plaintext Finished message would contain the message type code '14'. If the observed Finished message contains a hexadecimal value of '14' at the position where the plaintext Finished message would contain the message type code, the test shall be repeated three times in total. In case the value of '14' can be observed in all three tests it can be assumed that the Finished message has indeed been sent in plaintext and the test has to be regarded as 'failed'. Otherwise it has to be assumed that the observation of the value '14' has been due to chance and that the Finished message has indeed been sent encrypted. In that latter case the test shall be regarded as 'passed'.

High-Level Test Description
Perform a successful handshake using one of the accepted ciphersuites and verify that the Server Finished message is encrypted.
PASS

FCS_TLSS_EXT.1.2

287 The evaluator shall send a Client Hello requesting a connection for all mandatory and selected protocol versions in the SFR (e.g. by enumeration of protocol versions in a test client) and verify that the server denies the connection for each attempt.

High-Level Test Description
Using a Lightship developed TLS client, connect to the TOE and attempt to negotiate SSL 2.0, SSL 3.0, TLS 1.0 and any unsupported, but otherwise valid TLS protocol versions contained in the PP.
PASS

FCS_TLSS_EXT.1.3

288 If using ECDHE ciphers, the evaluator shall attempt a connection using an ECDHE ciphersuite and a configured curve. Using a packet analyser, verify that the key agreement parameters in the Key Exchange message are the ones configured. (Determining that the size matches the expected size for the configured curve is sufficient.) The evaluator shall repeat this test for each supported NIST Elliptic Curve and each supported Diffie-Hellman key size.

NOTE: ECDHE ciphersuites are not claimed by the TOE.

289 The evaluator shall attempt establishing connection using each claimed key establishment protocol (RSA, DH, ECDHE) with each claimed parameter (RSA key size, Diffie-Hellman parameters, supported curves) as selected in FCS_TLSS_EXT.1.3. For example, determining that the RSA key size matches the

claimed size is sufficient to satisfy this test. The evaluator shall ensure that each supported parameter combination is tested.

High-Level Test Description
Using a Lightship developed TLS client, connect to the TOE using a valid RSA DHE ciphersuite and verify that the public key size that comes back in the Server Key Exchange message matches the expected bit size for the chosen DH parameter. Using a Lightship developed TLS client, connect to the TOE using a valid pure RSA ciphersuite and verify that the certificate that comes back from the Server Certificate message matches the expected bit size.
PASS

290 Note that this testing can be accomplished in conjunction with other testing activities

5.2 Identification and Authentication (FIA)

5.2.1 FIA_X509_EXT.1/Rev X.509 Certificate Validation

5.2.1.1 TSS

291 The evaluator shall ensure the TSS describes where the check of validity of the certificates takes place, and that the TSS identifies any of the rules for extendedKeyUsage fields (in FIA_X509_EXT.1.1) that are not supported by the TOE (i.e. where the ST is therefore claiming that they are trivially satisfied). It is expected that revocation checking is performed when a certificate is used in an authentication step and when performing trusted updates (if selected). It is not necessary to verify the revocation status of X.509 certificates during power-up self-tests (if the option for using X.509 certificates for self-testing is selected).

Findings:	Section 6.1 of the [ST] for FIA_X509_EXT.1/Rev, FIA_X509_EXT.2, FIA_X509_EXT.3 indicates that the check of validity occurs at the time of import of the certificate. This is consistent with the fact that this TOE only offers a TLS server (which does not validate its own certificate except at import time).

292 The TSS shall describe when revocation checking is performed and on what certificates. If the revocation checking during authentication is handled differently depending on whether a full certificate chain or only a leaf certificate is being presented, any differences must be summarized in the TSS section and explained in the Guidance.

293 It is expected that revocation checking is performed when a certificate is used in an authentication step. It is expected that revocation checking is performed on both leaf and intermediate CA certificates when a leaf certificate is presented to the TOE as part of the certificate chain during authentication. Revocation checking of any CA certificate designated a trust anchor is not required. It is not sufficient to perform a revocation check of a CA certificate only when it is loaded onto the device.

Findings:	Section 6.1 of the [ST] for FIA_X509_EXT.1/Rev, FIA_X509_EXT.2, FIA_X509_EXT.3 indicates that revocation checking occurs at the time of import of the certificate. The check is performed on all certificates in the chain where a CRL has been configured. CA certificates are checked whenever a new leaf certificate is loaded.

5.2.1.2 Tests

294 The evaluator shall demonstrate that checking the validity of a certificate is performed when a certificate is used in an authentication step or when performing trusted updates (if FPT_TUD_EXT.2 is selected). It is not sufficient to verify the status of a X.509 certificate only when it is loaded onto the TOE. It is not necessary to verify the revocation status of X.509 certificates during power-up self-tests (if the option for using X.509 certificates for self-testing is selected). The evaluator shall perform the following tests for FIA_X509_EXT.1.1/Rev. These tests must be repeated for each distinct security function that utilizes X.509v3 certificates. For example, if the TOE implements certificate-based authentication with IPSEC and TLS, then it shall be tested with each of these protocols:

a. Test 1a: The evaluator shall present the TOE with a valid chain of certificates (terminating in a trusted CA certificate) as needed to validate the leaf certificate to be used in the function, and shall use this chain to demonstrate that the function succeeds. Test 1a shall be designed in a way that the chain can be 'broken' in Test 1b by either being able to remove the trust anchor from the TOEs trust store, or by setting up the trust store in a way that at least one intermediate CA certificate needs to be provided, together with the leaf certificate from outside the TOE, to complete the chain (e.g. by storing only the root CA certificate in the trust store)

   Test 1b: The evaluator shall then 'break' the chain used in Test 1a by either removing the trust anchor in the TOE's trust store used to terminate the chain, or by removing one of the intermediate CA certificates (provided together with the leaf certificate in Test 1a) to complete the chain. The evaluator shall show that an attempt to validate this broken chain fails.

High-Level Test Description
After running FIA_X509_EXT.3 which, in this TOE, showcases test 1a, we will proceed to remove the intermediate CA from the trust store. The TOE will react by removing the certificate in the certificate store which is no longer valid.
PASS

b. Test 2: The evaluator shall demonstrate that validating an expired certificate results in the function failing.

High-Level Test Description
Create an X.509 certificate with a 'notAfter' date in the past. Attempt to load this certificate into the TOE and show it is not accepted. Create an intermediate certificate with a 'notAfter' date in the past. Attempt to load this certificate into the TOE and show it is not accepted. Create a trust anchor with a 'notAfter' date in the past. Attempt to load this certificate into the TOE and show it is not accepted. Create a trust anchor certificate with a 'notAfter' date which is about to expire. Load this certificate into the TOE. Wait for the certificate to expire, then try to load a leaf certificate and show it is not accepted.
PASS

c. Test 3: The evaluator shall test that the TOE can properly handle revoked certificates-–conditional on whether CRL or OCSP is selected; if both are selected, then a test shall be performed for each method. The evaluator shall test revocation of the peer certificate and revocation of the peer intermediate CA certificate i.e. the intermediate CA certificate should be revoked by the root CA. The evaluator shall ensure that a valid certificate is used, and that the validation function succeeds. The evaluator then attempts the test with a certificate that has been revoked (for each method chosen in the selection) to ensure when the certificate is no longer valid that the validation function fails. Revocation checking is only applied to certificates that are not designated as trust anchors. Therefore, the revoked certificate(s) used for testing shall not be a trust anchor.

High-Level Test Description
Force the TOE to load a CRL which has no revoked certificates. Show that attempting to load the server certificate succeeds. Force the TOE to load a CRL containing a revoked server certificate. Show that attempting to load the server certificate fails. Force the TOE to load a CRL containing a revoked intermediate certificate. Show that attempting to load the server certificate fails.
PASS

d. Test 4: If OCSP is selected, the evaluator shall configure the OCSP server or use a man-in-the-middle tool to present a certificate that does not have the OCSP signing purpose and verify that validation of the OCSP response fails. If CRL is selected, the evaluator shall configure the CA to sign a CRL with a certificate that does not have the cRLsign key usage bit set, and verify that validation of the CRL fails.

High-Level Test Description
Modify the Intermediate CA to have a keyUsage which is missing the cRLSign policy. Force the TOE to load a CRL which has no revoked certificates. Show that attempting to load the server certificate fails.
PASS

e. Test 5: The evaluator shall modify any byte in the first eight bytes of the certificate and demonstrate that the certificate fails to validate. (The certificate will fail to parse correctly.)

High-Level Test Description
Attempt to load a mangled leaf certificate into the trust store and show that this fails.
PASS

f. Test 6: The evaluator shall modify any byte in the last byte of the certificate and demonstrate that the certificate fails to validate. (The signature on the certificate will not validate.)

High-Level Test Description
Attempt to load a leaf certificate into the trust store that has the last byte modified and show that this fails.
PASS

g. Test 7: The evaluator shall modify any byte in the public key of the certificate and demonstrate that the certificate fails to validate. (The hash of the certificate will not validate.)

High-Level Test Description
Attempt to load a leaf certificate into the trust store that has the last byte in the public key modulus modified and show that this fails.
PASS

295 The evaluator shall perform the following tests for FIA_X509_EXT.1.2/Rev. The tests described must be performed in conjunction with the other certificate services assurance activities, including the functions in FIA_X509_EXT.2.1/Rev. The tests for the extendedKeyUsage rules are performed in conjunction with the uses that require those rules. Where the TSS identifies any of the rules for extendedKeyUsage fields (in FIA_X509_EXT.1.1) that are not supported by the TOE (i.e. where the ST is therefore claiming that they are trivially satisfied) then the associated extendedKeyUsage rule testing may be omitted.

296 The goal of the following tests is to verify that the TOE accepts a certificate as a CA certificate only if it has been marked as a CA certificate by using basicConstraints with the CA flag set to True (and implicitly tests that the TOE correctly parses the basicConstraints extension as part of X509v3 certificate chain validation).

297 For each of the following tests the evaluator shall create a chain of at least three certificates: a self-signed root CA certificate, an intermediate CA certificate and a leaf (node) certificate. The properties of the certificates in the chain are adjusted as described in each individual test below (and this modification shall be the only invalid aspect of the relevant certificate chain).

a. Test 1: The evaluator shall ensure that at least one of the CAs in the chain does not contain the basicConstraints extension. The evaluator confirms that the TOE rejects such a certificate at one (or both) of the following points: (i) as part of the validation of the leaf certificate belonging to this chain; (ii) when attempting to add a CA certificate without the basicConstraints extension to the TOE's trust store (i.e. when attempting to install the CA certificate as one which will be retrieved from the TOE itself when validating future certificate chains).

High-Level Test Description
Load a CA certificate into the trust store which is missing the basicConstraints extension. Show that it fails to load.
PASS

b. Test 2: The evaluator shall ensure that at least one of the CA certificates in the chain has a basicConstraints extension in which the CA flag is set to FALSE. The evaluator confirms that the TOE rejects such a certificate at one (or both) of the following points: (i) as part of the validation of the leaf certificate belonging to this chain; (ii) when attempting to add a CA certificate with the CA flag set to FALSE to the TOE's trust store (i.e. when attempting to install the CA certificate as one which will be retrieved from the TOE itself when validating future certificate chains).

High-Level Test Description
Load a CA certificate into the trust store which has a basicConstraints of False extension. Show that it fails to load.
PASS

298 The evaluator shall repeat these tests for each distinct use of certificates. Thus, for example, use of certificates for TLS connection is distinct from use of certificates for trusted updates so both of these uses would be tested. But there is no need to repeat the tests for each separate TLS channel in FTP_ITC.1 and FTP_TRP.1/Admin (unless the channels use separate implementations of TLS).

Findings:	All distinct use of certificates have been covered.

5.2.2 FIA_X509_EXT.2 X.509 Certificate Authentication

5.2.2.1 TSS

299 The evaluator shall check the TSS to ensure that it describes how the TOE chooses which certificates to use, and any necessary instructions in the administrative guidance for configuring the operating environment so that the TOE can use the certificates.

Findings:	The TSS claims in section 6.1 of the [ST] for FIA_X509_EXT.1/Rev, FIA_X509_EXT.2, FIA_X509_EXT.3 that the administrative user manually installs and selects the certificate used by the TOE for each certificate. The [SUPP] in section 3.3.5 and section 3.3.6 provides the necessary guidance to administrators.

300 The evaluator shall examine the TSS to confirm that it describes the behaviour of the TOE when a connection cannot be established during the validity check of a certificate used in establishing a trusted channel. The evaluator shall verify that any distinctions between trusted channels are described. If the requirement that the administrator is able to specify the default action, then the evaluator shall ensure that the guidance documentation contains instructions on how this configuration action is performed.

Findings:	Section 6.1 of the [ST] for FIA_X509_EXT.1/Rev, FIA_X509_EXT.2, FIA_X509_EXT.3 states that if the connection to determine the certificate validity cannot be established, the TOE will accept the certificate based on the last known state.

5.2.2.2 Tests

301 The evaluator shall perform the following test for each trusted channel:

302 The evaluator shall demonstrate that using a valid certificate that requires certificate validation checking to be performed in at least some part by communicating with a

non-TOE IT entity. The evaluator shall then manipulate the environment so that the TOE is unable to verify the validity of the certificate, and observe that the action selected in FIA_X509_EXT.2.2 is performed. If the selected action is administrator-configurable, then the evaluator shall follow the guidance documentation to determine that all supported administrator-configurable options behave in their documented manner.

High-Level Test Description
Configure CRL sources to poll regularly. After configuration is complete, disconnect the CRL source from the environment. Show that when the TOE attempts to connect the CRL server to download the update, it fails. Show that the last state of the certificate is retained by the TOE.
PASS

5.2.3 FIA_X509_EXT.3 Extended: X509 Certificate Requests

5.2.3.1 TSS

303 If the ST author selects "device-specific information", the evaluator shall verify that the TSS contains a description of the device-specific fields used in certificate requests.

Findings:	The SFR in section 5.2 of the [ST] does not claim “device-specific information”.

5.2.3.2 Guidance Documentation

The evaluator shall check to ensure that the guidance documentation contains instructions on requesting certificates from a CA, including generation of a Certificate Request. If the ST author selects "Common Name", "Organization", "Organizational Unit", or "Country", the evaluator shall ensure that this guidance includes instructions for establishing these fields before creating the Certification Request.

Findings:	[SUPP] in section 3.3.4 provides some guidance to administrators to the fields which can be filled by administrators.

5.2.3.3 Tests

304 The evaluator shall perform the following tests:

a. Test 1: The evaluator shall use the guidance documentation to cause the TOE to generate a Certification Request. The evaluator shall capture the generated message and ensure that it conforms to the format specified. The evaluator shall confirm that the Certification Request provides the public key and other required information, including any necessary user-input information.

High-Level Test Description
Using the CLI interface, construct a new X.509 CSR and show that it contains the necessary information required by the SFR.
PASS

b. Test 2: The evaluator shall demonstrate that validating a response message to a Certification Request without a valid certification path results in the function failing. The evaluator shall then load a certificate or certificates as trusted CAs needed to validate the certificate response message, and demonstrate that the function succeeds.

High-Level Test Description
Attempt to load a signed certificate response without the associated trust chain. Show that the certificate fails to be imported. Load the trust anchor and any intermediate certificates into the trust store. Upload the signed certificate response and show that the attempt is successful.
PASS

5.3 Security management (FMT)

5.3.1 FMT_MOF.1/Functions Management of security functions behaviour

5.3.1.1 TSS

305 For distributed TOEs see chapter 2.4.1.1. There are no specific requirements for non-distributed TOEs.

Findings:	The TOE is not a distributed TOE.

5.3.1.2 Tests

306 Test 1 (if ‘transmission of audit data to external IT entity’ is selected from the second selection together with 'modify the behaviour of' in the first selection): The evaluator shall try to modify all security related parameters for configuration of the transmission protocol for transmission of audit data to an external IT entity without prior authentication as security administrator (by authentication as a user with no administrator privileges or without user authentication at all). Attempts to modify parameters without prior authentication should fail. According to the implementation no other users than the Security Administrator might be defined and without any user authentication the user might not be able to get to the point where the attempt to modify the security related parameters can be executed. In that case it shall be demonstrated that access control mechanisms prevent execution up to the step that can be reached without authentication as Security Administrator.

Findings:	This function is not selected.

307 Test 2 (if ‘transmission of audit data to external IT entity’ is selected from the second selection together with 'modify the behaviour of' in the first selection): The evaluator shall try to modify all security related parameters for configuration of the transmission protocol for transmission of audit data to an external IT entity with prior authentication as security administrator. The effects of the modifications should be confirmed.

308 The evaluator does not have to test all possible values of the security related parameters for configuration of the transmission protocol for transmission of audit data to an external IT entity but at least one allowed value per parameter.

Findings:	This function is not selected.

309 Test 1 (if 'handling of audit data' is selected from the second selection together with 'modify the behaviour of' in the first selection): The evaluator shall try to modify all security related parameters for configuration of the handling of audit data without prior authentication as security administrator (by authentication as a user with no administrator privileges or without user authentication at all). Attempts to modify parameters without prior authentication should fail. According to the implementation no other users than the Security Administrator might be defined and without any user authentication the user might not be able to get to the point where the attempt can be executed. In that case it shall be demonstrated that access control mechanisms prevent execution up to the step that can be reached without authentication as Security Administrator. 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/LocSpace.

Findings:	This function is not selected.

310 Test 2 (if 'handling of audit data' is selected from the second selection together with 'modify the behaviour of' in the first selection): The evaluator shall try to modify all security related parameters for configuration of the handling of audit data with prior authentication as security administrator. The effects of the modifications should be confirmed. 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/LocSpace.

311 The evaluator does not necessarily have to test all possible values of the security related parameters for configuration of the handling of audit data but at least one allowed value per parameter.

Findings:	This function is not selected.

312 Test 1 (if 'audit functionality when Local Audit Storage Space is full' is selected from the second selection together with 'modify the behaviour of' in the first selection): The evaluator shall try to modify the behaviour when Local Audit Storage Space is full without prior authentication as security administrator (by authentication as a user with no administrator privileges or without user authentication at all). This attempt should fail. According to the implementation no other users than the Security Administrator might be defined and without any user authentication the user might not be able to get to the point where the attempt can be executed. In that case it shall be demonstrated that access control mechanisms prevent execution up to the step that can be reached without authentication as Security Administrator.

Findings:	This function is not selected.

313 Test 2 (if 'audit functionality when Local Audit Storage Space is full' is selected from the second selection together with 'modify the behaviour of' in the first selection): The evaluator shall try to modify the behaviour when Local Audit Storage Space is full with prior authentication as security administrator. This attempt should be successful. The effect of the change shall be verified.

314 The evaluator does not necessarily have to test all possible values for the behaviour when Local Audit Storage Space is full but at least one change between allowed values for the behaviour.

Findings:	This function is not selected.

315 Test 3 (if in the first selection 'determine the behaviour of' has been chosen together with for any of the options in the second selection): The evaluator shall try to determine the behaviour of all options chosen from the second selection without prior authentication as security administrator (by authentication as a user with no administrator privileges or without user authentication at all). This can be done in one test or in separate tests. The attempt(s) to determine the behaviour of the selected functions without administrator authentication shall fail. According to the implementation no other users than the Security Administrator might be defined and without any user authentication the user might not be able to get to the point where the attempt can be executed. In that case it shall be demonstrated that access control mechanisms prevent execution up to the step that can be reached without authentication as Security Administrator.

High-Level Test Description
As a privileged administrator, attempt to view the log transfer settings using the CLI and show the attempt is successful. As an unprivileged administrator, attempt to view the log transfer settings using the CLI and show the attempt is unsuccessful. As a privileged administrator, attempt to view the log transfer settings using the Web GUI and show the attempt is successful. Copy the URL for the log transfer settings. As an unprivileged administrator, attempt to view the log transfer settings using the CLI and show the attempt is unsuccessful. Paste the URL for the log transfer settings into the Web UI and show that direct navigation is not successful.
PASS

316 Test 4 (if in the first selection 'determine the behaviour of' has been chosen together with for any of the options in the second selection): The evaluator shall try to determine the behaviour of all options chosen from the second selection with prior authentication as security administrator. This can be done in one test or in separate tests. The attempt(s) to determine the behaviour of the selected functions with administrator authentication shall be successful.

Findings:	Please refer to previous test case.

5.3.2 FMT_MTD.1/CryptoKeys Management of TSF Data

5.3.2.1 TSS

317 For distributed TOEs see chapter 2.4.1.1. There are no specific requirements for non-distributed TOEs.

Findings:	The TOE is not a distributed TOE.

5.3.2.2 Tests

318 The evaluator shall try to perform at least one of the related actions (modify, delete, generate/import) without prior authentication as security administrator (either by authentication as a non-administrative user, if supported, or without authentication at all). Attempts to perform related actions without prior authentication should fail. According to the implementation no other users than the Security Administrator might be defined and without any user authentication the user might not be able to get to the point where the attempt to manage cryptographic keys can be executed. In that case it shall be demonstrated that access control mechanisms prevent execution up to the step that can be reached without authentication as Security Administrator.

319 The evaluator shall try to perform at least one of the related actions with prior authentication as security administrator. This attempt should be successful.

High-Level Test Description
As a non-privileged administrator, attempt to generate a new public/private key pair and show the operation is not permitted. Use of the privileged administrator to execute the same is done in FIA_X509_EXT.1/Rev.
PASS

6 Evaluation Activities for SARs

6.1 ADV: Development

6.1.1 Basic Functional Specification (ADV_FSP.1)

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

Findings:	The [SUPP], [ADMIN] and [CLI] documents provide a comprehensive treatment of each of the security-relevant TSFI. Note that CLI-based commands are covered in basic terms within the [SUPP] in various sections and more completely within [CLI]. Web UI elements are covered specifically within the context of the Common Criteria within [SUPP] and with more details in [ADMIN]. The ST and the AGD comprise the functional specification. If the test in [SD] cannot be completed because the [ST] or the [SUPP] (or [ADMIN] or [CLI]) is incomplete, then the functional specification is not complete and observations are required. During the evaluator's use of the product and its interfaces (the Web GUI, local serial console CLI and remote SSH CLI), there were no areas that were deficient at the conclusion of the evaluation. Please refer to the preceding Assurance Activities for more details.

321 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.

Findings:	Please refer to the previous discussion.

322 The evaluator shall examine the interface documentation to develop a mapping of the interfaces to SFRs.

Findings:	This was conducted by the evaluator and coded into the test plan. In the test plan, this mapping is provided by the sub-table with the headings: 'Interface', 'TSFI Command(s)' and 'Description' which maps the specific CLI command or GUI component to a specific management interface under a specific SFR test case.

6.2 AGD: Guidance Documents

6.2.1 Operational User Guidance (AGD_OPE.1)

323 The evaluator shall ensure the Operational guidance documentation is 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.

Findings:	This TOE is destined for the NIAP Product Compliance List (PCL) which mandates the public distribution of the Common Criteria supplement [SUPP] document. Therefore, there is a reasonable guarantee that administrators and users are aware

of the existence and role of this supplementary guide in establishing and maintaining the evaluated configuration. The additional documents which are hyperlinked in the [SUPP] in section 1.3 are available to the public without any paywall.

324 The evaluator shall ensure that the Operational guidance is provided for every Operational Environment that the product supports as claimed in the Security Target and shall adequately address all platforms claimed for the TOE in the Security Target.

Findings: The provided operational guidance covers all claimed platforms in the [ST]. The operational environment is reasonably generic as to be useful in most circumstances.

325 The evaluator shall ensure that the Operational guidance contains 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.

Findings: There is only one cryptographic engine included in the TOE and therefore warnings about additional engines are unnecessary.

326 The evaluator shall ensure the Operational guidance makes it clear to an administrator which security functionality and interfaces have been assessed and tested by the EAs.

Findings: The operational guidance – in the form of the [SUPP] – describes the evaluated functionality in section 1.6 (excluded functionality) and more importantly, section 3 (scope of included functionality).

327 In addition the evaluator shall ensure that the following requirements are also met.

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 instructions that describe at least one method of validating 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.

Findings: (a) There is only one cryptographic engine included in the TOE and therefore warnings about additional engines are unnecessary.

(b) The documentation provides instructions for verifying updates to the TOE by means of a published hash. This information can be found in section 2.1 of the

	[SUPP]. This information includes the instructions for obtaining the TOE and the published hash. In this section, upgrades are obtained from the vendor's software distribution website and placed onto the TOE using a documented procedure. The published hash is provided at the same location when the TOE software is manually downloaded. When the TOE software is downloaded by the TOE itself, the hash appears in the updater_logs as per the documented procedure in section 2.1 of the [SUPP]. (c) The operational guidance – in the form of the [SUPP] – describes the evaluated functionality in section 1.6.

6.2.2 Preparative Procedures (AGD_PRE.1)

328 The evaluator shall examine the Preparative procedures to ensure they 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).

Findings:	The [SUPP] in section 1.5 provides information on the necessary components that must reside in the operational environment. The vious hardware guides (referred to as [HW170] and [HWx95] in this document) provide information on the appropriate physical environment needed to fulfil the TOE's needs. For the virtual appliance, the necessary hypervisor is documented in the [VAPP] document.

329 The evaluator shall examine the Preparative procedures to ensure they are provided for every Operational Environment that the product supports as claimed in the Security Target and shall adequately address all platforms claimed for the TOE in the Security Target.

Findings:	The provided operational guidance in [SUPP] and the links provided in section 1.3 of [SUPP] cover all claimed platforms in the [ST].

330 The evaluator shall examine the preparative procedures to ensure they include instructions to successfully install the TSF in each Operational Environment.

Findings:	The provided operational guidance in [SUPP] and the links provided in section 1.3 of [SUPP] cover instructions suitable to install the TOE into an appropriate operational environment.

331 The evaluator shall examine the preparative procedures to ensure they include instructions to manage the security of the TSF as a product and as a component of the larger operational environment.

Findings:	The [SUPP], [ADMIN] and [CLI] documentation provides extensive information on managing the security of the TOE as an individual product. Additional best practice guidance provided within those documents helps impart a culture of secure manageability within a larger operational environment.

332 In addition, the evaluator shall ensure that the following requirements are also met.

333 The preparative procedures must

a) include instructions to provide a protected administrative capability; and

b) identify TOE passwords that have default values associated with them and instructions shall be provided for how these can be changed.

Findings:	The entire [SUPP] document is designed to ensure the administrator is aware of how to configure the TOE to provide a protected administrative capability. The TOE has default passwords out of the box. When the TOE is configured for the first time, the passwords for the accounts described in section 3.2 of the [SUPP] are changed as part of the initial configuration process workflow.

7 Vulnerability Assessment

334 The evaluator shall examine the documentation outlined below provided by the developer to confirm that it contains all required information. This documentation is in addition to the documentation already required to be supplied in response to the EAs listed previously.

335 The developer shall provide documentation identifying the list of software and hardware components[3] that compose the TOE. Hardware components should identify at a minimum the processors used by the TOE. Software components include applications, the operating system and other major components that are independently identifiable and reusable (outside the TOE) such as a web server and protocol or cryptographic libraries. This additional documentation is merely a list of the name and version number of the components, and will be used by the evaluators in formulating hypotheses during their analysis.

Findings:	The evaluator collected this information from the developer which was used to feed into the Type 1 Flaw Hypotheses search (below).

336 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 Appendix A.3. The evaluator shall perform vulnerability analysis in accordance with Appendix A.2. The results of the analysis shall be documented in the report according to Appendix A.3.

Findings:	The following sources of public vulnerabilities were considered in formulating the specific list of flaws to be investigated by the evaluators, as well as to reference in directing the evaluators to perform key-word searches during the evaluation of the TOE. Hypothesis sources for public vulnerabilities were: Vendor security advisories: https://tools.cisco.com/security/center/publicationListing.x NIST National Vulnerability Database (can be used to access CVE and US-CERT databases identified below): https://web.nvd.nist.gov/view/vuln/search CVE Details: https://www.cvedetails.com/ US-CERT: http://www.kb.cert.org/vuls/html/search SecurITeam Exploit Search: www.securiteam.com Tenable Network Security: http://nessus.org/plugins/index.php?view=search Tipping Point Zero Day Initiative: http://www.zerodayinitiative.com/advisories Offensive Security Exploit Database: https://www.exploit-db.com/

[3] In this sub-section the term “components” refers to parts that make up the TOE. It is therefore distinguished from the term “distributed TOE components”, which refers to the parts of a TOE that are present in one physical part of a distributed TOE. Each distributed TOE component will therefore generally include a number of the hardware and software components that are referred to in this sub-section: for example, each distributed TOE component will generally include hardware components such as processors and software components such as an operating system and libraries.

Rapid7 Vulnerability Database: https://www.rapid7.com/db/vulnerabilities

OpenSSL Vulnerabilities: https://www.openssl.org/news/vulnerabilities.html

OpenSSH Release Notes: https://www.openssh.com/releasenotes.html

Google

Type 1 Hypothesis searches were last conducted on July 22, 2021 and included the following search terms (many version details have been removed in this public document):

Email Security Appliance

C190, C195, C390, C395, C690, C690X, C695, C695F, C100v, C300v, C600v

Cisco AsyncOS 13.0

ESXi 6.0

Glass web server

OpenSSH

OpenSSL

TLS

SSH v2

TCP

The evaluation team determined that, based on these searches, no residual vulnerabilities exist that are exploitable by attackers with Basic Attack Potential.

There are no type-2 hypotheses identified for the NDcPP.

The evaluation team developed Type 3 flaw hypotheses in accordance with Sections A.1.3, A.1.4, and A.2, and no residual vulnerabilities exist that are exploitable by attackers with Basic Attack Potential.

The evaluation team developed Type 4 flaw hypotheses in accordance with Sections A.1.3, A.1.4, and A.2, and no residual vulnerabilities exist that are exploitable by attackers with Basic Attack Potential.