E-Series & EF-Series with SANtricity OS 11.70

Assurance Activity Report

Version 1.1

November 2021

Document prepared by

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	NetApp, Inc.
TOE	NetApp E-Series & EF-Series with SANtricity OS 11.70 Build (Management software version): R2 08.70.00.02
Security Target	NetApp E-Series & EF-Series with SANtricity OS 11.70 Security Target, v1.0, November 2021
Protection Profile	collaborative Protection Profile for Network Devices, v2.2e, 20200323 (NDcPP)

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	Evaluation Activities for Network Device cPP, v2.2, 20191204 (NDcPP-SD)
Interpretations	**NDcPP v2.2e+20200323**
	TD 0527 – Updates to Certificate Revocation Testing (FIA_X509_EXT.1) ***This TD applies to the TOE and AAs will be adhered to.***
	TD 0528 - NIT Technical Decision for Missing EAs for FCS_NTP_EXT.1.4

	This TD does not apply to the TOE as FCS_NTP_EXT.1 is not claimed.
	TD 0536 - NIT Technical Decision for Update Verification Inconsistency ***This TD applies to the TOE and AAs will be adhered to.***
	TD 0537 - NIT Technical Decision for Incorrect reference to FCS_TLSC_EXT.2.3 ***This TD does not apply to the TOE as FCS_TLSC_EXT.2 is not claimed.***
	TD 0538 - NIT Technical Decision for Outdated link to allowed-with list ***This TD does not apply to the TOE as this is for Protection Profile.***
	TD 0546 - NIT Technical Decision for DTLS – Clarification of Application Note 63 ***This TD does not apply to the TOE because FCS_DTLS is not claimed.***
	TD 0547 - NIT Technical Decision for Clarification on Developer Disclosure of AVA_VAN ***This TD applies to the TOE and AAs will be adhered to.***
	TD 0555 - NIT Technical Decision for RFC Reference Incorrect in TLSS Test ***This TD applies to the TOE and AAs will be adhered to.***
	TD 0556 – NIT Technical Decision for RFC 5077 question ***This TD applies to the TOE and AAs will be adhered to.***
	TD 0563 – NIT Technical Decision for Clarification of audit date information ***This TD applies to the TOE and AAs will be adhered to.***
	TD 0564 – NIT Technical Decision for Vulnerability Analysis Search Criteria ***This TD applies to the TOE and AAs will be adhered to.***
	TD 0569 – NIT Technical Decision for Session ID Usage Conflict in FCS_DTLSS_EXT.1.7 ***This TD does not apply to the TOE because FCS_DTLSS is not claimed.***
	TD 0570 – NIT Technical Decision for Clarification about FIA_AFL.1 ***This TD applies to the TOE and AAs will be adhered to.***
	TD 0571 – NIT Technical Decision for Guidance on how to handle FIA_AFL.1

	This TD applies to the TOE and AAs will be adhered to.
	TD 0572 – NIT Technical Decision for Restricting FTP_ITC.1 to only IP address identifiers ***This TD applies to the TOE and AAs will be adhered to.***
	TD 0580 – NIT Technical Decision for clarification about use of DH14 in NDcPPv2.2e ***This TD applies to the TOE and AAs will be adhered to.***
	TD 0581 – NIT Technical Decision for Elliptic curve-based key establishment and NIST SP 800-56A rev3 ***This TD applies to the TOE and AAs will be adhered to.***
	TD 0591 – NIT Technical Decision for Virtual TOEs and hypervisors ***This TD does not apply to the TOE as it is not a virtual TOE or hypervisor.***
	TD 0592 – NIT Technical Decision for Local Storage of Audit Records ***This TD does not apply to the TOE because the affected requirements are not claimed.***

2 TOE Details

2.1 Overview

1 The TOE is a network device that provides networked storage for dedicated, high-bandwidth applications like data analytics, video surveillance, and disk-based backup that need simple, fast, reliable SAN storage.

2.2 Claimed Models

Table 3: Claimed Models

Model	CPU	Max Capacity	Max Drives
E2812 (DE212C)*	Intel Xeon D-15xx (Broadwell)	576TB	48 HDD/SSD
E2824 (DE224C)	Intel Xeon D-15xx (Broadwell)	173TB	96 HDD/SSD
E2860 (DE460C)	Intel Xeon D-15xx (Broadwell)	2.16PB	180 HDD/SSD
E5724 (DE224C)	Intel Xeon D-15xx (Broadwell)	345TB	192 HDDs / 120 SSDs
E5760 (DE460C)	Intel Xeon D-15xx (Broadwell)	4.8PB	480 HDDs / 120 SSDs
EF280	Intel Xeon D-15xx (Broadwell)	1.5PB	96 SSDs
EF570	Intel Xeon D-15xx (Broadwell)	1.8PB	120 SSDs
EF600	Intel Xeon D-21xx (Skylake)	368TB	24 SSDs
EF300	Intel Xeon D-21xx (Skylake)	368TB	24 SSDs

* Disk shelf shown in parentheses. The disk shelf is an enclosure that contains the system shelf (E-series controller) and hot-serviceable drive trays.

2.3 Test Platform Equivalency

3 The TOE is a Network Device that includes both hardware and software components. The evaluator determined the platform equivalency based on SD Appendix A.7.2 Guidance for Network Devices and selected the following subset (please refer to Table 5: Hardware Requirements) of models for evaluation.

2.3.1 Platform Equivalency Rationale

4 The following table provides a description of evaluator considerations of each of the factors that affect equivalency between TOE model variations and across operating environments.

Table 4: Platform Equivalency Consideration Factors

Factor	Dependencies	Rationale
Platform/Hardware Dependencies	The evaluator selected two models for testing based on the following factors : • E-Series vs EF-Series • CPU Architecture	The E-Series models use Intel Xeon D-15xx (Broadwell), whereas EF-Series models use Intel Xeon D-21xx (Skylake). There are no Security Functional (SF)-related differences between the models.
Differences in TOE Software Binaries	Identical for all the models	Each instance of NetApp E-Series & EF-Series with SANtricity OS 11.70 Build (Management software version): R2 08.70.00.02 is produced from the same source code and compiled to a binary targeting a specfic CPU architecture.
Differences in Libraries Used to Provide TOE Functionality	Identical for all the models	There is no difference in the libraries used between the models.
TOE Management Interface Differences	Identical for all the models	There is no difference in the management interfaces between the models. All the models support all the following interfaces as identified in the ST: • CLI • Web GUI • REST API
TOE Functional Differences	Identical for all the models	There are no SF-related differences between models. All the Security Functionalities identified in the ST are implemented consistently across all the models.

2.3.2 Test Platform Subsets

5 Following appliances were used for the testing based on the platform equivalency details provided in the above section.

Table 5: Hardware Requirements

Identification	Series	Model Name	CPU Architecture	CPU
TOE - A	E-Series Hybrid flash arrays	E2860 (DE460C)	X86-64	Intel Xeon D-15xx (Broadwell)
TOE - B	EF-Series All flash arrays	EF600	X86-64	Intel Xeon D-21xx (Skylake)

2.4 Reference Documents

Table 6: List of Reference Documents

Ref	Document
[ST]	NetApp E-Series & EF-Series with SANtricity OS 11.70 Security Target, v1.0, November 2021
[PP]	collaborative Protection Profile for Network Devices, v2.2e, 20200323 (NDcPP)
[SD]	Evaluation Activities for Network Device cPP, v2.2, 20191204 (NDcPP-SD)
[SUPP]	NetApp E-Series & EF-Series with SANtricity OS 11.70 Common Criteria Guidance Supplement, v0.8 September 2021
[SETUP]	NetApp E-Series E5724, EF570, EF280, E2812, E2824, DE212C and DE224C shelves Installation and Setup Instructions, 2020 NetApp E-Series E2860, E5760, and DE460C Shelves Installation and Setup Instructions, 2020 NetApp EF-Series EF300 and EF600 Installation and Setup Instructions, 2020
[CLI]	SANtricity System Manager SMcli Online Help for SANtricity, October 2020
[REST]	NetApp SANtricity Web Services API Documentation, Version: 05.20

2.5 Summary of SFRs

Table 7: 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_TLSC_EXT.1	TLS Client 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/ManualUpdate	Management of security functions behaviour
FMT_MOF.1/Functions	Management of security functions behaviour
FMT_MTD.1/CoreData	Management of TSF Data
FMT_MTD.1/CryptoKeys	Management of TSF Data
FMT_SMF.1	Specification of Management Functions
FMT_SMR.2	Restrictions on Security Roles

Requirement	Title
FPT_SKP_EXT.1	Protection of TSF Data (for reading of all pre-shared, symmetric and private keys)
FPT_APW_EXT.1	Protection of Administrator Passwords
FPT_TST_EXT.1	TSF Testing (Extended)
FPT_TUD_EXT.1	Trusted Update
FPT_STM_EXT.1	Reliable Time Stamps
FTA_SSL_EXT.1	TSF-initiated Session Locking
FTA_SSL.3	TSF-initiated Termination
FTA_SSL.4	User-initiated Termination
FTA_TAB.1	Default TOE Access Banners
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

6 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:	The evaluator checked and identified from ST TSS section 6.1.1 that for the administrative task of generating, importing and deletion of cryptographic keys, the TOE logs both the operation performed by the security administrator and the relevant key reference.

7 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

8 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:	Audit events and format are presented in section 3.1 of the [SUPP]. There are extensive samples of the expected audit messages provided in section 3 of the [SUPP] and the pertinent information is highlighted in each of those samples.

9 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. 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. In the end, the vendor provided a more comprehensive guidance "supplement" document in the form of [SUPP]. However, AAs for guidance were performed combining all the Guidance documents listed in the **Table 6: List of Reference Documents** of this document.

3.1.1.3 Tests

10 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:	**Pass** The evaluator confirmed the TOE produces all of the required audit events in conjunction with the applicable test activity and verified that it included all of the required fields and matched the reference samples provided in [SUPP]

11 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.

12 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

3.1.2.1 TSS & Guidance Documentation

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

3.1.2.2 Tests

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

14 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

15 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 TSS, in section 6.1.3 of the [ST] claims that the audit data is transferred using TLS (as described by FCS_TLSC_EXT.1) in real-time.

16 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:	Section 6.1.3 of the TSS in the [ST] indicates that the number of audit log entries stored locally can be user-defined. When the audit log is full, the oldest log entries are overwritten. There can be a lag in the system which might permit slightly *more* records than configured, though after discussions with the vendor, this is due to how the system operates (log overwriting is asynchronous). Only authorized administrators may view audit records and no capability to modify the audit records is provided. An administrator may delete audit logs.

17 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:	Section 6.1.3 of the TSS in the [ST] indicates that the TOE is a standalone TOE that stores audit data locally.

18 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:	Section 6.1.3 of the TSS in the [ST] indicates that the number of audit log entries stored locally can be user-defined. When the audit log is full, the oldest log entries are overwritten. There can be a lag in the system which might permit slightly *more* records than configured, though after discussions with the vendor, this is due to how the system operates (log overwriting is asynchronous).

19 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:	As detailed in section 6.1.3 of the [ST], the transmission is done in real-time.

20 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.

21 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

22 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:	This information can be found in section 2.1 of the [SUPP]. The TOE communicates with a syslog-capable server over TLS 1.2. The ciphersuites, curves and certificate requirements are described.

23 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 2.1 of the [SUPP] describes the relationship: “The event is propagated to the TOE data store and is also simultaneously emitted to any audit log servers that are configured for receipt of log events.” This is consistent with section 6.1.3 of the [ST] that the audit data is transferred using TLS (as described by FCS_TLSC_EXT.1) in real-time.

24 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:	The TOE only claims “overwrite” of old audit log data and therefore additional description of this functionality is unnecessary.

3.1.3.3 Tests

25 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.

Findings:	This test is performed in conjunction with FTP_ITC.1.

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).

Findings:	**Pass**

The evaluator configured the maximum audit records held by the TOE to 2000 events. The evaluator observed the oldest audit event then performed actions to generate additional audits. Once the threshold of 2000 events had been exceeded, the evaluator queried the audit log and confirmed that the oldest events had been overwritten.

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 functionality.

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

26 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: The ST provides details on key sizes in section 6.2 for asymmetric cryptographic keys. The TOE implements RSA 2048 and ECC with Curves P-256 and P-521. In both cases, the key generation schemes are used for TLS authentication.

3.2.1.2 Guidance Documentation

27 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: There are no configuration options available for key generation schemes, sizes, protocols. It is enabled by default as RSA 2048 for server key generation and ECC for TLS handshake.

3.2.1.3 Tests

28 **(NIAP TD0580 applied)** Note: The following tests require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on factory products.

Key Generation for FIPS PUB 186-4 RSA Schemes

29 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*.

30 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

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

Key Generation for Elliptic Curve Cryptography (ECC)

FIPS 186-4 ECC Key Generation Test

32 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

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

Key Generation for Finite-Field Cryptography (FFC)

34 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.

35 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

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

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

37 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.

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

39 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.

40 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

41 for each FFC parameter set and key pair.

42 FFC Schemes using "safe-prime" groups

43 Testing for FFC Schemes using safe-prime groups is done as part of testing in CKM.2.1.

Findings:	The vendor uses CAVP certificate C1977 for RSA key generation. The vendor uses CAVP certificate C1977 for EC key generation. These are described in the Security Target in Table 4.

3.2.2 FCS_CKM.2 Cryptographic Key Establishment

3.2.2.1 TSS

44 **(NIAP TD0580 applied)** 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.

45 Te 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
ECDH	FCS_IPSEC_EXT.1	Authentication Server

46 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, section 6.2.2 identifies the supported key establishment schemes. The TOE uses ECDHE for TLS key exchange. The TOE acts as both a sender when operating in a TLS server capacity and receiver when operating in a TLS client capacity. This corresponds to the key generation scheme identified in FCS_CKM.1.1. The evaluator checked and confirmed that Section 6.2.2 FCS_CKM.1.2 provides sufficient information on the key establishment scheme supported by the TOE, relevant SFRs and services in association with para 64 of the SD.

3.2.2.2 Guidance Documentation

47 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:	The ECC cryptographic key establishment scheme is enabled by default in its only possible configuration and it is the only allowed scheme for Key establishment operation. Section 2.2 of [SUPP] lists the supported TLS ciphersuites for TLS handshake.

3.2.2.3 Tests

Key Establishment Schemes

48 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

49 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

50 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.

51 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.

52 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.

53 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.

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

Validity Test

55 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.

56 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).

57 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.

RSA-based key establishment

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.

FFC Schemes using "safe-prime" groups

The evaluator shall verify the correctness of the TSF's implementation of safeprime groups 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 safe-

prime groups. This test must be performed for each safe-prime group that each protocol uses.

Findings:	The vendor uses CAVP certificate C1977 for RSA key generation. The vendor uses CAVP certificate C1977 for EC key generation. The vendor uses CAVP certificate C1977 for KAS operations in key agreement schemes. These are described in the Security Target in Table 4.

3.2.3 FCS_CKM.4 Cryptographic Key Destruction

3.2.3.1 TSS

58 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:	All relevant keys are described in table 14 in section 6.5.1 which includes their origin and their storage location. Keys live in both persistent Java Keystore as well as in RAM. RAM-based keys are plaintext and the Java Keystore RSA key is deleted when a new certificate is imported or when certificates are removed. The tables do appear to describe all relevant keys. The TOE claims cryptographic channels covering TLS for trusted channels. The various session keys are consistent with the TLS protocol implementation.

59 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 mechanism by which the TOE destroys plaintext keys in non-volatile memory is described in the ST in Table 14. ST Section 6.5.1 FPT_SKP_EXT.1 says “Keys are protected as described in ST Table 14: Private Keys. In all cases, plaintext keys cannot be viewed through an interface designed specifically for that purpose.

[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.

Key	Generation/ Algorithm	Storage	Zeroization
TLS Private Key	RSA (2048 bits)	Persistent – Java Keystore	The TLS private key is deleted when a new certificate is imported or when certificates are removed. The TOE will destroy the abstraction that represents the key via the JVM garbage collector.
DH Parameters used for TLS	ECDH (secp256r1, secp512r1)	RAM - plaintext	JVM garbage collector when no longer required.
AES key used for TLS	AES-128 AES-256	RAM - plaintext	JVM garbage collector when no longer required.

60 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 TOE claims in FCS_CKM.4 destruction of reference to the key directly followed by a request for garbage collection for volatile memory. The TOE implements a JVM interface which is consistent with the method described. For non-volatile memory, destruction of the underlying reference of the key in the Java Keystore is performed which is consistent with the use of the Java Keystore for persistent key storage.

61 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:	The TOE does not claim any keys are stored in non-plaintext format.

62 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 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:	No such information is conveyed in the ST TSS. There are no obvious circumstances that would prevent conformance to the described mechanism.

63 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:	The TOE does not claim this selection.

3.2.3.2 Guidance Documentation

64 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.

65 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:	There are no obvious circumstances where delayed or prevented key destruction can occur. The [ST] in section 6.5.1 describes that the TOE employs garbage collection to deal with logical key destruction.

3.2.3.3 Tests

None

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

3.2.4.1 TSS

66 The evaluator shall examine the TSS to ensure it identifies the key size(s) and mode(s) supported by the TOE for data encryption/decryption.

Findings:	The ST, section 6.2.4 identifies that the TOE supports symmetric encryption and decryption capabilities using 128 and 256 bit AES in GCM mode for TLS.

3.2.4.2 Guidance Documentation

67 The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected mode(s) and key size(s) defined in the Security Target supported by the TOE for data encryption/decryption.

Findings:	There is no configuration available to enable a specific mode or key size of the supported algorithm (AES) as the TOE is restricted to allow only AES-128 and AES-256 in GCM mode during TLS handshake.

[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).

3.2.4.3 Tests

AES-CBC Known Answer Tests

68 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.

69 **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.

70 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.

71 **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.

72 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.

73 **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].

74 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.

75 **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].

76 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

77 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.

78 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

79 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]
```

80 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.

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

AES-GCM Test

82 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.

83 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.

84 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.

85 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.

AES-CTR Known Answer Tests

86 There are four Known Answer Tests (KATs) described below. 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-CTR-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.

94 There is no need to test the decryption engine.

Findings:	The vendor uses CAVP certificate C1977 for AES in GCM mode. This is described in the Security Target in Table 4.

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

3.2.5.1 TSS

95 The evaluator shall examine the TSS to determine that it specifies the cryptographic algorithm and key size supported by the TOE for signature services.

Findings:	The ST, section 6.2.5 identifies that the TOE provides cryptographic signature generation and verification services using: a) RSA Signature Algorithm with key size of 2048 bit, b) ECDSA Signature Algorithm with NIST curves P-256 and P-521. The RSA and ECDSA signature verification services are used in the TLS protocols.

3.2.5.2 Guidance Documentation

96 The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected cryptographic algorithm and key size defined in the Security Target supported by the TOE for signature services.

Findings:	There are no configuration options available for TOE signature services. It is enabled by default to support both RSA 2048 and ECDSA with NIST curves P-256 and P-521 in the evaluated configuration.

3.2.5.3 Tests

ECDSA Algorithm Tests

ECDSA FIPS 186-4 Signature Generation Test

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

the evaluator shall use the signature verification function of a known good implementation.

ECDSA FIPS 186-4 Signature Verification Test

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

RSA Signature Algorithm Tests

Signature Generation Test

99 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.

100 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.

Signature Verification Test

101 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.

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

Findings:	The vendor uses CAVP certificate C1977 for RSA signature generation and verification. The vendor uses CAVP certificate C1977 for EC signature generation and verification. These are described in the Security Target in Table 4.

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

3.2.6.1 TSS

103 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:	This information is documented in section 6.2.6 of the ST. The TSS section specifies that the TOE provides cryptographic hashing services using SHA-1, SHA-256 and SHA384. SHS is implemented in the following parts of the TSF: a) TLS; and b) Hashing of passwords in non-volatile storage.

3.2.6.2 Guidance Documentation

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

Findings:	Cryptographic hash configuration is enabled by default, and matches the ST requirements.

3.2.6.3 Tests

105 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.

106 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

107 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

108 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

109 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 $1 \leq i \leq m$. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Selected Long Messages Test - Byte-oriented Mode

110 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 $1 \leq i \leq m/8$. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Pseudorandomly Generated Messages Test

111 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:	The vendor uses CAVP certificate C1977 for SHA1 and SHA2 cryptographic hashing. These are described in the Security Target in Table 4.

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

3.2.7.1 TSS

112 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:	This information is documented in section 6.2.7 of the ST. The TSS section specifies that the TOE provides keyed-hashing message authentication services using HMAC-SHA-1, HMAC-SHA-256, and HMAC-SHA-384. HMAC is implemented in the TLS protocol. The details of the key length, block size, hash function and output MAC length are found in table 13 of the ST.

3.2.7.2 Guidance Documentation

113 The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the values used by the HMAC function: key length, hash function used, block size, and output MAC length used defined in the Security Target supported by the TOE for keyed hash function.

Findings:	Cryptographic keyed hash configuration is enabled by default, and matches the ST requirements.

3.2.7.3 Tests

114 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:	The vendor uses CAVP certificate C1977 for keyed hashing leveraging SHA1 and SHA2 from the previous SFR component. These are described in the Security Target in Table 4.

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

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

3.2.8.1 TSS

116 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:	The DRBG type has been documented in the ST Section 6.2.9. The TOE implements a Hash(SHA-256)_DRBG that is seeded from Intel RDRAND that provides full 256 bits of Entropy. Additional detail is provided in the proprietary Entropy Description.

3.2.8.2 Guidance Documentation

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

Findings:	There are no additional instructions required to configure the RNG functionality. It is preconfigured and enabled by default.

3.2.8.3 Tests

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

119 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).

120 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.

121 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:	The vendor uses CAVP certificate C1977 for RBG operations. This is described in the Security Target in Table 4.

3.3 Identification and Authentication (FIA)

3.3.1 FIA_AFL.1 Authentication Failure Management

3.3.1.1 TSS

122 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:	The ST describes authentication failure handling in section 6.3.5 of the TSS. Specifically, each defined administrative interface except the local console will enforce authentication failures in a uniform way. Locked accounts are locked out until an administrator-defined time limit expires. Each supported method of remote administrative actions is actually described in section 6.3.2 of the ST: direct local console access, remote access over the web GUI and remote access over the REST API.

123 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:	The local console is not subjected to the lock out mechanisms.

3.3.1.2 Guidance Documentation

124 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:	This is found in [SUPP] section 2.4. It describes configuration of lockout time, and maximum login attempts. It offers guidance on configuration for secure operation.

125 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:	As described in section 2.6 of the [SUPP], remote (SSH) administration is not permitted and disabled by default. Local access through serial cable assures continued access to admin, with local admin password, which in turn can be used to unlock the remote admin accounts.

3.3.1.3 Tests

126 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.

Findings:	**Pass** The evaluator configured the authentication failure threshold to 3 attempts and a lockout for 10 minutes. The evaluator then attempted to log in with the incorrect password 3 times via the Web GUI and confirmed that the user was locked out. The evaluator then attempted to log in from a separate IP using the correct credentials and confirmed the user was still locked out, showing the lockout was not IP based. The evaluator then waited 8 minutes and attempted to log in with the correct credentials and confirmed that the user was still locked out. The evaluator then waited until the timer reached 10 minutes and successfully logged in with the correct credentials. This test was then repeated via the REST API and as an LDAP user. The CLI has its own lockout mechanism that requires the user to reidentify after 5 failed attempts. The evaluator attempted to log in via the CLI with the incorrect password 5 times and confirmed that the log in session was closed, forcing the user to reidentify.

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).

Findings:	The TOE only claims time-based lockout.

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.

Findings:	This was performed in conjunction with test 1 above.

3.3.2 FIA_PMG_EXT.1 Password Management

3.3.2.1 TSS

127 The evaluator shall examine the TSS to determine that it contains the lists of the supported special character(s) and minimum and maximum number of charters supported for administrator passwords.

Findings:	The ST Section 6.3.1 contains the list of the supported special characters that the password can be composed of. This section also mentions that the length of the password is configurable by the Administrator to between 1 and 30 characters.

3.3.2.2 Guidance Documentation

128 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:	A description on how to configure passwords is provided in [SUPP] in section 2.5. Guidance is provided to users on creating suitable strong passwords.

3.3.2.3 Tests

129 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.

b) Test 2: The evaluator shall compose passwords that do not meet the requirements in some way. For each password, the evaluator shall verify that the TOE does not support the password. While the evaluator is not required (nor is it feasible) to test all possible compositions of passwords, the evaluator shall ensure that the TOE enforces the allowed characters and the minimum length listed in the requirement and justify the subset of those characters chosen for testing.

Findings:	**Pass** The evaluator changed the minimum password length to 15 characters then set the user's password to a 15 character length password and confirmed the user could log in. The evaluator then changed the minimum password length to 8 characters and confirmed that an attempt to set the user's password to a 7 character length was rejected. The evaluator then attempted to change the password to an 8 character length and confirmed that it was accepted and user could log in.

3.3.3 FIA_UIA_EXT.1 User Identification and Authentication

3.3.3.1 TSS

130 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 6.3.2 and section 6.3.3 of the ST collectively provide the requested information. In section 6.3.2 of the ST, the various interfaces are enumerated with the interface dispositions (local vs. remote). In section 6.3.3, each interface is described in terms of what constitutes a successful logon. This process differs depending on whether a non-TOE authentication server is involved.

131 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:	The ST identifies the actions available prior to I&A in section 6.3.2 FIA_UIA_EXT.1. These are limited to displaying the warning banner and storage services. The warning banner is displayed for all interactive interfaces (the CLI and the Web GUI) and storage services are always available regardless of presented administrative interface.

132 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.

133 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

134 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:	[SUPP] section 2.6 describes available authentication interfaces and login methods. No preparatory steps are necessary. The steps to achieve a successful login are provided.

3.3.3.3 Tests

135 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.

Findings:	**Pass** The evaluator confirmed that a user could successfully log in via the Web GUI using the correct password. The evaluator then attempted to log in via the Web GUI using an incorrect password and confirmed access to the TOE was not granted. The evaluator repeated this test for the CLI, REST API and as an LDAP user.

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.

Findings:	**Pass** The evaluator confirmed that the only service available on the Web GUI before log in was viewing the log in banner.

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.

Findings:	**Pass** The evaluator confirmed that the only service available on the CLI before log in was viewing the log in banner.

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

136 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.

Findings:	Assurance activities for this SFR are covered under FIA_UIA_EXT.1.

3.3.5 FIA_UAU.7 Protected Authentication Feedback

3.3.5.1 TSS

None.

3.3.5.2 Guidance

137 The evaluator shall examine the guidance documentation to determine that any necessary preparatory steps to ensure authentication data is not revealed while entering for each local login allowed.

Findings:	[SUPP] section 2.6 describes available authentication interfaces and login methods. No preparatory steps are necessary.

3.3.5.3 Tests

138 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.

Findings:	**Pass** The evaluator confirmed that when logging into the local console that there was not password feedback echoed back to the user.

3.4 Security management (FMT)

3.4.1 General requirements for distributed TOEs

3.4.1.1 TSS

139 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

140 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

141 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 Management of Security Functions Behaviour

3.4.2.1 TSS

142 For distributed TOEs see chapter 3.4.1.1. There are no specific requirements for non-distributed TOEs.

Findings:	The TOE is not a distributed TOE.

3.4.2.2 Guidance Documentation

143 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:	[SUPP] section 2.8 provides the update procedures and the error code that might be encountered for an update failure. The [SUPP] describes that "there are no security functions that may cease to operate during the update."

144 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

145 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.

146 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:	**Pass** The evaluator attempted to update the TOE via the Web GUI as an unauthorized user and confirmed the attempt was rejected. The evaluator then repeated this test via the REST API and observed the same result.

	Attempts to update the TOE via an authorized user were performed in conjunction with FPT_TUD_EXT.1

3.4.3 FMT_MTD.1/CoreData Management of TSF Data

3.4.3.1 TSS

147 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.

148 If the 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 administrative functions available prior to I&A are described in FIA_UIA_EXT.1 above. According to the described functionality in section 6.3.2 of the ST, the TSF shall require each administrative user to be successfully identified and authenticated before allowing any other TSF-mediated actions on behalf of that administrative user. The section 6.3.7 of the ST mentions that the TOE has a trust store where root CA and intermediate CA certificates can be stored. The TOE restricts the ability to access trust store to Security Administrators.

3.4.3.2 Guidance Documentation

149 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:	The evaluator considered the following TSF-data manipulating functions implemented in response to the requirements of the CPP: Ability to modify the size of the local audit log via the REST API (REST API documentation under "POST /devmgr/v2/storage-systems/{system-id}/auditlog/config"); Ability to configure minimum password length (under "Change local user password settings" in the SANtricity Help Dashboard for System Manager) and in section 2.5 of the [SUPP]; Ability to modify user lockout parameters via the REST API (REST API documentation under "POST /devmgr/storage-systems/{system-id}/settings/lockout") and in section 2.4 of the [SUPP]; Ability to unlock administrative accounts as described in section 2.6 of the [SUPP]; Managing user session timeouts (under "Managing session timeouts" in the SANtricity Help Dashboard for System Manager) and in section 2.16 of the [SUPP]; Configuring the login banner (under "Configure login banner" in the SANtricity Help Dashboard for System Manager) and in section 2.18 of the [SUPP];

Configuring syslog receivers (section 2.1 and section 2.2 of the [SUPP]); and

Managing the certificate trust store (section 2.2 of the [SUPP]).

The entire management plane is constructed from REST API calls and each of the REST API calls are documented as to which user role is permitted to perform the tasks. These can be seen in the [REST] documentation for each call and for each REST verb associated with the call.

150 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: Section 2.9 of [SUPP] specifies how the administrator can manage Trusted CA certificates including how to import a designated trust anchor.

3.4.3.3 Tests

No separate testing for FMT_MTD.1/CoreData is required unless one of the management functions has not already been exercised under any other SFR.

3.4.4 FMT_SMF.1 Specification of Management Functions

151 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)

152 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. 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: The information was found in section 6.4.5 of the ST and each function is tagged with the interface in which it is available for security managers to use.

153 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 information was found in section 6.4.5 of the ST that CLI is used to administer the TOE locally.

154 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.

3.4.4.2 Guidance Documentation

155 See section 3.4.4.1.

3.4.4.3 Tests

156 The evaluator tests management functions as part of testing the SFRs identified in section 3.4.4. 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:	No separate testing is required.

3.4.5 FMT_SMR.2 Restrictions on security roles

3.4.5.1 TSS

157 The evaluator shall examine the TSS to determine that it details the TOE supported roles and any restrictions of the roles involving administration of the TOE.

Findings:	The information was found in section 6.4.6 of the ST and it describes all the supported roles of the TOE and privileges/restrictions of the roles involving administration of the TOE .

3.4.5.2 Guidance Documentation

158 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:	Section 2.11 of the [SUPP] indicates that the TOE is capable of both remote and local administration. The local console is quite limited and operates for rescue purposes only.

3.4.5.3 Tests

159 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:	**Pass** The evaluator tested all interfaces through testing of the other test activities.

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

160 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:	The storage of the symmetric keys and private keys are described in section 6.5.1 Table 14 of the ST. The information is not protected at rest, but no interfaces are provided to access these materials.

3.5.2 FPT_APW_EXT.1 Protection of Administrator Passwords

3.5.2.1 TSS

161 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:	Administrative passwords are described in section 6.5.2 of the ST. The information is protected at rest using Salted SHA-256 hash. No interfaces are provided to access this material directly.

3.5.3 FPT_TST_EXT.1 TSF testing

3.5.3.1 TSS

162 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:	Detailed self-tests are described in section 6.5.3 of the ST. These tests include CPU and BIOS self-tests, software integrity test and FIPS 140-2 KATs. These tests are argued by the ST author as being sufficient. These tests appear to be sufficient from the perspective of the evaluator: (a) the hardware upon which the TOE software is running can be relied upon because the CPU and memory are appropriately tested using read and write operations; (b) the boot loader's image is verified implying that the boot loader code has not been modified accidentally; (c) trusted administrators (as per A.TRUSTED_ADMINISTRATOR) are expected to load binary firmware into the TOE that meets the published hash verification check and thus, post-bootloader, the software/firmware package can be relied upon to be in a known trusted state; (d) the cryptographic module undergoes FIPS 140-2 known answer tests which provide reliability for critical cryptographic functionality.

163 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

164 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:	Section 2.12 of the [SUPP] describes the self-tests which occur in the TOE. When the errors occur, the TOE will log an audit event which can direct the administrator to perform the documented recovery procedures.

165 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

166 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.

167 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.

168 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.

Findings:	**Pass** The evaluator reset the TOE and observed that the system integrity self-tests were executed via the audit log. The evaluator also observed the feedback from the local console during the TOE reset and observed that the TOE properly initialized.

169 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

170 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:	The active version can be viewed using the Web UI as per section 6.5.4 of the ST. Updates are applied immediately. The TOE provides the ability to “stage” an update, but this is not the same as delayed activation since the staged firmware is not actually installed.

171 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:	The ST describes the web GUI mechanism for upgrading the TOE in section 6.5.4. Authenticated administrators can optionally use the non-interactive REST API to deploy firmware upgrades as well. The TOE relies on trusted administrators performing a review of a published hash of the firmware prior to being installed.

172 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 5.3.5 of the ST claims that the TOE does not support automatic updates therefore this requirement does not apply.

173 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.

174 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:	The section ST 6.5.4 claims that **t**he administrator validates the firmware update by generating a SHA-256 hash of the downloaded update file and comparing the resulting hash to the published SHA256 hash on the NetApp download portal.

3.5.4.2 Guidance Documentation

175 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:	The current version of the software can be queried by visiting “Controller software and firmware upgrades” in the Web UI as described in the SANtricity Help Dashboard for System Manager.

176 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:	Section 2.13 of the [SUPP] directs users to compare the cryptographic hash found on the NetApp customer Support site to the downloaded firmware binary package. This is consistent with the TSS in the ST.

177 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:	The source for the published hash can be found by visiting the NetApp customer Support site as described in section 2.13 of the [SUPP].

178 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 Security Administrator; 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.

179 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.

180 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 is not a distributed TOE.

3.5.4.3 Tests

181 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.

Findings:	**Pass** The evaluator queried the version of the TOE then calculated the SHA256 hash of the update file and confirmed it matched the published value. The evaluator then applied the update to the TOE and queried the version again to confirm the update was successful.

b) 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).

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 does not support digital signatures.

c) 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).

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 Security Administrator 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) The evaluator uses a legitimate update and tries to perform verification of the hash value without providing the published hash value on 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 TOE does not verify the published hash. All published hashes are compared by the Security Administrator offline.

182 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.

183 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 TOE only supports manual updates.

184 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

185 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:	The evaluator found the required information in section 6.5.5 of the ST. The TOE has a built-in time source which is manually set by the Security Administrator during initial TOE configuration. The various functions that make use of the time are also disclosed.

3.5.5.2 Guidance Documentation

186 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 [SUPP] in section 2.13 describes how to set the system time. It provides information on what to do if there is an error while attempting to synchronize the two controllers' time. NTP is not claimed.

3.5.5.3 Tests

187 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.

Findings:	**Pass** The evaluator queried the time on the TOE then attempted to change the time. The evaluator queried the time on the TOE again and confirmed that the time had successfully changed.

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.

Findings:	The TOE does not claim NTP.

188 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.

Findings:	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 TSS

189 The evaluator shall examine the TSS to determine that it details whether local administrative session locking or termination is supported and the related inactivity time period settings.

Findings:	Section 6.6.1 of the ST claims the local session (CLI) supports idle timeout termination. The timeout value is set to fifteen minutes by default but may be configured by the Security Administrator.

3.6.1.2 Guidance Documentation

190 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:	Section 2.14 of the [SUPP] claims that the local session supports idle timeout termination. The idle timer is set using the same mechanism as for the remote management session.

3.6.1.3 Tests

191 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.

Findings:	**Pass** The evaluator set the timeout value to 15 minutes then logged in to the CLI. The evaluator waited 15 minutes and confirmed that the user was successfully logged out. The evaluator then repeated this test with a 20 minute inactivity threshold.

3.6.2 FTA_SSL.3 TSF-initiated Termination

3.6.2.1 TSS

192 The evaluator shall examine the TSS to determine that it details the administrative remote session termination and the related inactivity time period.

Findings:	Section 6.6.2 of the ST claims the remote session (Web UI) supports idle timeout termination. The timeout value is set to thirty minutes by default but may be configured by the Security Administrator.

3.6.2.2 Guidance Documentation

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

Findings:	Section 2.15 of the [SUPP] describes that idle timeout termination is employed for the remote administrative sessions. The timer itself can be set using the REST API as described in section 2.16 of the [SUPP] or via the Web UI as described under "Managing session timeouts" in the SANtricity Help Dashboard for System Manager.

3.6.2.3 Tests

194 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.

Findings:	**Pass** The evaluator set the timeout value to 15 minutes then logged in to the Web GUI. The evaluator waited 15 minutes and confirmed that the user was successfully logged out. The evaluator then repeated this test with a 20 minute inactivity threshold.

3.6.3 FTA_SSL.4 User-initiated Termination

3.6.3.1 TSS

195 The evaluator shall examine the TSS to determine that it details how the local and remote administrative sessions are terminated.

Findings:	Section 6.6.3 of the ST claims that the administrative users may terminate their own sessions at any time.

3.6.3.2 Guidance Documentation

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

Findings:	Section 2.16 of the [SUPP] describes how to terminate both the local and remote administrative interactive sessions.

3.6.3.3 Tests

197 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.

Findings:	**Pass** The evaluator logged into the TOE via the CLI and typed ‘q’ to quit the session and confirmed that the user was successfully logged out.

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.

Findings:	**Pass** The evaluator logged in to the TOE via the Web GUI and clicked the logout button and confirmed that the user was successfully logged out.

3.6.4 FTA_TAB.1 Default TOE Access Banners

3.6.4.1 TSS

198 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:	The TSS indicates the methods of access in section 6.6.4 of the ST. The TOE banner is indicated on each of those interactive mechanisms. These include CLI and Web GUI. Note that the REST API is not considered an interactive mechanism due to its transactional API nature. This is consistent with Application Note 36 in the PP.

3.6.4.2 Guidance Documentation

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

Findings:	Configuring the login banner is described under “Configure login banner” in the SANtricity Help Dashboard for System Manager and in section 2.17 of the [SUPP].

3.6.4.3 Tests

200 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.

Findings:	**Pass**

	The evaluator logged into the TOE and changed the log in banner. The evaluator then logged out and attempted to log in via the Web GUI and CLI and confirmed the banner was displayed before logging in.

3.7 Trusted path/channels (FTP)

3.7.1 FTP_ITC.1 Inter-TSF trusted channel

3.7.1.1 TSS

201 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:	Trusted channels are described in section 6.7 of the ST. The TOE provides a channel between syslog and LDAP server entities. Each channel is categorized according to how the communication can be initiated. The channels are described with their respective cryptographic protocols (TLS for syslog and LDAP). These protocols are consistent to the cryptographic protocols listed in Section 6.2.10 in the ST. The method of non-TOE endpoint authentication is described in section 6.2.10 of the ST.

3.7.1.2 Guidance Documentation

202 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 [SUPP] describes in section 2.18 how to configure the underlying cryptographic protocols which protect the trusted channels. In addition, when the links are unintentionally broken, the administrator may have to restart them using the “Test All” functionality built into the Web UI and REST API interfaces.

3.7.1.3 Tests

203 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.

Findings:	**Pass** The evaluator tested connections via a trusted channel to an audit server and LDAP server via test 2 below (LDAP) and FCS_TLSC_EXT.1.1 Test 1 (syslog).

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.

Findings:	**Pass** The evaluator attempted to log in as an LDAP user and observed that the TOE successfully communicated with the LDAP and the communications were encrypted.

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

The developer 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.

Findings:	**Pass** The evaluator tested connections via a trusted channel to an audit server and LDAP server via test 2 above (LDAP) and FCS_TLSC_EXT.1.1 Test 1 (syslog).

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 network link 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.

Findings:	**Pass** The evaluator performed 2 tests, one short disconnect and one long disconnect). The evaluator logged into the TOE as an LDAP user and generated audit messages to be sent to the syslog server. The evaluator confirmed communications to each server were encrypted then physically disconnected the servers from an intermediate switch. The evaluator then attempted to log in as an LDAP user and generate audit messages and confirmed the TOE attempted to communicate with the server but was unsuccessful and the channel data was not sent in plaintext. The evaluator then reconnected the servers and attempted to log in as the LDAP user and generate audit

	messages again and confirmed that the communications were restored and the channel data was sent encrypted.

204 Further assurance activities are associated with the specific protocols.

205 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.

3.7.2 FTP_TRP.1/Admin Trusted Path

3.7.2.1 TSS

206 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:	The ST section 6.7.2 claims that the TOE provides the following methods of remote administration using secure HTTP protocol: a) Web GUI over HTTPS per FCS_HTTPS_EXT.1.1 b) REST API over HTTPS per FCS_HTTPS_EXT.1.1

3.7.2.2 Guidance Documentation

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

Findings:	Section 2.19 of the [SUPP] describes that the TOE operates as a TLS server to provide access to the Web UI or the REST API. In the [REST], under section “What are the supported browsers”, the types and versions of web clients are described.

3.7.2.3 Tests

208 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.

Findings:	This test is performed in conjunction with FCS_TLSS_EXT.1.1 Test 1.

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

Findings:	This test is performed in conjunction with FCS_TLSS_EXT.1.1 Test 1.

209 Further assurance activities are associated with the specific protocols.

210 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

No optional requirement have been selected for this evaluation.

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

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

Findings:	The ST in section 6.2.8 provides a description as to how the TOE conforms with RFC 2818. Specifically, section 6.2.8 provides a rationale for the various sections in RFC 2818 where implementations might stray from the requirements set down in the PP. The rationale is acceptable.

5.1.1.2 Guidance Documentation

The evaluator shall examine the guidance documentation to verify it instructs the Administrator how to configure TOE for use as an HTTPS client or HTTPS server.

Findings:	Note the TOE only functions as an HTTPS server. Section 2.3 of the [SUPP] specifies the configuration needed to put the TOE into the evaluated configuration. Section 2.19 of the [SUPP] also states that the TOE is configured with a default self-signed certificate however the user can install their own CA signed certificate. Section 2.9 of the [SUPP] includes instructions for installing such certificates.

5.1.1.3 Tests

212 This test is now performed as part of FIA_X509_EXT.1/Rev testing.

213 Tests are performed in conjunction with the TLS evaluation activities.

214 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_TLSC_EXT.1 Extended: TLS Client Protocol

5.1.2.1 TSS

FCS_TLSC_EXT.1.1

215 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 include those listed for this component.

Findings:	In section 6.2.10 of the ST TSS, the TLS client ciphersuites are referred back to the SFR definition in section 5.3.2 of the ST. These ciphersuites are consistent with the permissible set defined in the SFR. These ciphersuites are consistent with the claimed cryptographic components from the various FCS_COP.1.

FCS_TLSC_EXT.1.2

216 The evaluator shall ensure that the TSS describes the client's method of establishing all reference identifiers from the administrator/application-configured reference identifier, including which types of reference identifiers are supported (e.g. application-specific Subject Alternative Names) and whether IP addresses and wildcards are supported.

Findings:	References identifiers for the syslog server and LDAP are defined in section 6.2.10 of the ST TSS as being configured by the admin using the Web GUI or REST API and must be an IP or a DNS. Both CN and SANs of the stated types are supported. Certificate pinning is claimed as not being supported and wildcards are claimed as being supported.

217 Note that where a TLS channel is being used between components of a distributed TOE for FPT_ITT.1, the requirements to have the reference identifier established by the user are relaxed and the identifier may also be established through a "Gatekeeper" discovery process. The TSS should describe the discovery process and highlight how the reference identifier is supplied to the "joining" component. Where the secure channel is being used between components of a distributed TOE for FPT_ITT.1 and the ST author selected attributes from RFC 5280, the evaluator shall ensure the TSS describes which attribute type, or combination of attributes types, are used by the client to match the presented identifier with the configured identifier. The evaluator shall ensure the TSS presents an argument how the attribute type, or combination of attribute types, uniquely identify the remote TOE component; and the evaluator shall verify the attribute type, or combination of attribute types, is sufficient to support unique identification of the maximum supported number of TOE components.

Findings:	The TOE is not a distributed TOE.

218 If IP addresses are supported in the CN as reference identifiers, the evaluator shall ensure that the TSS describes the TOE's conversion of the text representation of the IP address in the CN to a binary representation of the IP address in network byte order. The evaluator shall also ensure that the TSS describes whether canonical format (RFC 5952 for IPv6, RFC 3986 for IPv4) is enforced.

Findings:	The ST TSS Section 6.2.10 claims that the TOE supports IPv4 as a reference identifier, however if an IPv4 address is used in the X.509 certificate, then a SAN is required. If a SAN is available and does not match the reference identifier, then the verification fails and the channel is terminated.

FCS_TLSC_EXT.1.4

219 The evaluator shall verify that TSS describes the Supported Elliptic Curves/Supported Groups Extension and whether the required behaviour is performed by default or may be configured.

Findings:	Section 6.2.10 of the ST TSS claims the Supported Elliptic Curves NIST curves P256 and P512. The non-TOE server can choose to negotiate the elliptic curve from this set for any of the mutually negotiable elliptic curve ciphersuites. The behaviour is enabled by default and cannot be configured.

5.1.2.2 Guidance Documentation

FCS_TLSC_EXT.1.1

220 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.

Findings:	There is no additional configuration required; the TOE conforms to the required configuration by default.

FCS_TLSC_EXT.1.2

221 The evaluator shall verify that the AGD guidance includes instructions for setting the reference identifier to be used for the purposes of certificate validation in TLS.

Findings:	The [SUPP] in section 2.2 describes that the LDAP and syslog reference identifiers can be provided in the URL fields when configuring those components.

FCS_TLSC_EXT.1.4

222 If the TSS indicates that the Supported Elliptic Curves/Supported Groups Extension must be configured to meet the requirement, the evaluator shall verify that AGD guidance includes configuration of the Supported Elliptic Curves/Supported Groups Extension.

Findings:	There is no additional configuration required; the TOE conforms to the required configuration by default.

5.1.2.3 Tests

FCS_TLSC_EXT.1.1

223 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).

Findings:	**Pass** The evaluator successfully established TLS connections from the TOE using each of the claimed ciphersuites in the [ST].

224 Test 2: The evaluator shall attempt to establish the connection using a server with a server certificate that contains the Server Authentication purpose in the extendedKeyUsage field and verify that a connection is established. The evaluator will then verify that the client rejects an otherwise valid server certificate that lacks the Server Authentication purpose in the extendedKeyUsage field and a connection is not established. Ideally, the two certificates should be identical except for the extendedKeyUsage field.

Findings:	**Pass** The evaluator attempted the successful connection for this test in conjunction with test 1 above. The evaluator then attempted a TLS connection from the TOE to a server presenting a certificate without the Server Authentication extendedKeyUsage and verified that the connection was not accepted by the TOE.

225 Test 3: The evaluator shall send a server certificate in the TLS connection that does not match the server-selected ciphersuite (for example, send an ECDSA certificate while using the TLS_RSA_WITH_AES_128_CBC_SHA ciphersuite). The evaluator shall verify that the TOE disconnects after receiving the server's Certificate handshake message.

Findings:	**Pass** The evaluator attempted a TLS connection from the TOE to a server that selected an RSA ciphersuite but presented an ECDSA certificate and verified that the connection was not accepted by the TOE.

226 Test 4: The evaluator shall perform the following 'negative tests':

a) The evaluator shall configure the server to select the TLS_NULL_WITH_NULL_NULL ciphersuite and verify that the client denies the connection.

Findings:	**Pass** The evaluator attempted a TLS connection from the TOE to a server that selects the TLS_NULL_WITH_NULL_NULL ciphersuite and verified that the connection was not accepted by the TOE.

b) Modify the server's selected ciphersuite in the Server Hello handshake message to be a ciphersuite not presented in the Client Hello handshake message. The evaluator shall verify that the client rejects the connection after receiving the Server Hello.

Findings:	**Pass** The evaluator attempted a TLS connection from the TOE to a server that selects the TLS_RSA_WITH_NULL_NULL_MD5 ciphersuite and verified that the connection was not accepted by the TOE.

c) [conditional]: If the TOE presents the Supported Elliptic Curves/Supported Groups Extension the evaluator shall configure the server to perform an ECDHE or DHE key exchange in the TLS connection using a non-supported curve/group (for example P-192) and shall verify that the TOE disconnects after receiving the server's Key Exchange handshake message.

Findings:	**Pass** The evaluator attempted a TLS connection from the TOE to a server that uses secp192r1 for the key exchange and verified that the connection was not accepted by the TOE.

227 Test 5: The evaluator performs the following modifications to the traffic:

a) Change the TLS version selected by the server in the Server Hello to a non-supported TLS version and verify that the client rejects the connection.

Findings:	**Pass**

	The evaluator attempted a TLS connection from the TOE to a server that presented TLS version 1.6 (0x0307) in the Server Hello and verified that the connection was not accepted by the TOE.

b) [conditional]: If using DHE or ECDH, modify the signature block in the Server's Key Exchange handshake message, and verify that the handshake does not finished successfully, and no application data flows. This test does not apply to cipher suites using RSA key exchange. If a TOE only supports RSA key exchange in conjunction with TLS, then this test shall be omitted.

Findings:	**Pass** The evaluator attempted a TLS connection from the TOE to a server that modified a byte in the Server Key Exchange message and verified that the connection was not accepted by the TOE.

228 Test 6: The evaluator performs the following 'scrambled message tests':

a) Modify a byte in the Server Finished handshake message and verify that the handshake does not finish successfully and no application data flows.

Findings:	**Pass** The evaluator attempted a TLS connection from the TOE to a server that modified a byte in the Server Finished message and verified that the connection was not accepted by the TOE.

b) Send a garbled message from the server after the server has issued the ChangeCipherSpec message and verify that the handshake does not finish successfully and no application data flows.

Findings:	**Pass** The evaluator attempted a TLS connection from the TOE to a server that sent a garbled message after the ChangeCipherSpec message and verified that the connection was not accepted by the TOE.

c) Modify at least one byte in the server's nonce in the Server Hello handshake message and verify that the client rejects the Server Key Exchange handshake message (if using a DHE or ECDHE ciphersuite) or that the server denies the client's Finished handshake message.

Findings:	**Pass** The evaluator attempted a TLS connection from the TOE to a server that sent a modified nonce in the Server Hello message and verified that the connection was not accepted by the TOE.

FCS_TLSC_EXT.1.2

229 Note that the following tests are marked conditional and are applicable under the following conditions:

a) For TLS-based trusted channel communications according to FTP_ITC.1 where RFC 6125 is selected, tests 1-6 are applicable.

or

b) For TLS-based trusted path communications according to FTP_TRP where RFC 6125 is selected, tests 1-6 are applicable

or

c) For TLS-based trusted path communications according to FPT_ITT.1 where RFC 6125 is selected, tests 1-6 are applicable. Where RFC 5280 is selected, only test 7 is applicable.

Note that for some tests additional conditions apply.

IP addresses are binary values that must be converted to a textual representation when presented in the CN of a certificate. When testing IP addresses in the CN, the evaluator shall follow the following formatting rules:

- IPv4: The CN contains a single address that is represented a 32-bit numeric address (IPv4) is written in decimal as four numbers that range from 0-255 separated by periods as specified in RFC 3986.

- IPv6: The CN contains a single IPv6 address that is represented as eight colon separated groups of four lowercase hexadecimal digits, each group representing 16 bits as specified in RFC 4291. Note: Shortened addresses, suppressed zeros, and embedded IPv4 addresses are not tested.

Findings:	The TOE is not a distributed TOE.

230 The evaluator shall configure the reference identifier according to the AGD guidance and perform the following tests during a TLS connection:

a) Test 1 [conditional]: The evaluator shall present a server certificate that contains a CN that does not match the reference identifier and does not contain the SAN extension. The evaluator shall verify that the connection fails. The evaluator shall repeat this test for each identifier type (e.g. IPv4, IPv6, FQDN) supported in the CN. When testing IPv4 or IPv6 addresses, the evaluator shall modify a single decimal or hexadecimal digit in the CN.

Remark: Some systems might require the presence of the SAN extension. In this case the connection would still fail but for the reason of the missing SAN extension instead of the mismatch of CN and reference identifier. Both reasons are acceptable to pass Test 1.

Findings:	**Pass** The evaluator attempted a TLS connection from the TOE to a server that presented a certificate with an invalid CN for the IP and no SAN extension and verified that the connection was not accepted by the TOE. The evaluator then repeated this test with the server using an invalid DNS value in the CN.

b) Test 2 [conditional]: The evaluator shall present a server certificate that contains a CN that matches the reference identifier, contains the SAN extension, but does not contain an identifier in the SAN that matches the reference identifier. The evaluator shall verify that the connection fails. The evaluator shall repeat this test for each supported SAN type (e.g. IPv4, IPv6, FQDN, URI). When testing IPv4 or IPv6 addresses, the evaluator shall modify a single decimal or hexadecimal digit in the SAN.

Findings:	**Pass** The evaluator attempted a TLS connection from the TOE to a server that presented a certificate with a valid CN for the IP and an invalid value in the SAN extension and verified that the connection was not accepted by the TOE. The evaluator then repeated this test with the server using a valid DNS value in the CN and an invalid DNS value in the SAN.

c) Test 3 [conditional]: If the TOE does not mandate the presence of the SAN extension, the evaluator shall present a server certificate that contains a CN that matches the reference identifier and does not contain the SAN extension. The evaluator shall verify that the connection succeeds. The evaluator shall repeat this test for each identifier type (e.g. IPv4, IPv6, FQDN) supported in the CN. If the TOE does mandate the presence of the SAN extension, this Test shall be omitted.

Findings:	**Pass** The evaluator attempted a TLS connection from the TOE to a server that presented a certificate with a valid CN for the DNS and no SAN extension and verified that the connection was accepted by the TOE. Note this test is not applicable for IP values since the TOE mandates the SAN for IP identifiers.

d) Test 4 [conditional]: The evaluator shall present a server certificate that contains a CN that does not match the reference identifier but does contain an identifier in the SAN that matches. The evaluator shall verify that the connection succeeds. The evaluator shall repeat this test for each supported SAN type (e.g. IPv4, IPv6, FQDN, SRV).

Findings:	**Pass** The evaluator attempted a TLS connection from the TOE to a server that presented a certificate with a valid CN for the IP and a valid value in the SAN extension and verified that the connection was accepted by the TOE. The evaluator then repeated this test with the server using a valid DNS value in the CN and a valid DNS value in the SAN.

e) Test 5 [conditional]: The evaluator shall perform the following wildcard tests with each supported type of reference identifier that includes a DNS name (i.e. CN-ID with DNS, DNS-ID, SRV-ID, URI-ID):

    1) [conditional]: The evaluator shall present a server certificate containing a wildcard that is not in the left-most label of the presented identifier (e.g. foo.*.example.com) and verify that the connection fails.

Findings:	**Pass**

	The evaluator attempted a TLS connection from the TOE to a server presenting a certificate with a wildcard not in the left-most label of the DNS value (foo.*.example.com) and verified that the TOE did not accept the connection.

2) [conditional]: The evaluator shall present a server certificate containing a wildcard in the left-most label (e.g. *.example.com). The evaluator shall configure the reference identifier with a single left-most label (e.g. foo.example.com) and verify that the connection succeeds, if wildcards are supported, or fails if wildcards are not supported. The evaluator shall configure the reference identifier without a left-most label as in the certificate (e.g. example.com) and verify that the connection fails. The evaluator shall configure the reference identifier with two left-most labels (e.g. bar.foo.example.com) and verify that the connection fails. (Remark: Support for wildcards was always intended to be optional. It is sufficient to state that the TOE does not support wildcards and observe rejected connection attempts to satisfy corresponding assurance activities.)

Findings:	**Pass** The evaluator configured the TLS server to present a certificate with a wildcard in the left-most label of the DNS (*.example.com) and attempted connections to it from the TOE with the following reference identifiers: foo.example.com, example.com, and foo.services.example.com. The evaluator confirmed that the TOE accepted the foo.example.com connection and rejected the others.

f) Test 6 [conditional]: If IP addresses are supported, the evaluator shall present a server certificate that contains a CN that matches the reference identifier, except one of the groups has been replaced with an asterisk (*) (e.g. CN=192.168.1.* when connecting to 192.168.1.20, CN=2001:0DB8:0000:0000:0008:0800:200C:* when connecting to 2001:0DB8:0000:0000:0008:0800:200C:417A). The certificate shall not contain the SAN extension. The evaluator shall verify that the connection fails. The evaluator shall repeat this test for each supported IP address version (e.g. IPv4, IPv6).

Remark: Some systems might require the presence of the SAN extension. In this case the connection would still fail but for the reason of the missing SAN extension instead of the mismatch of CN and reference identifier. Both reasons are acceptable to pass Test 6.

Findings:	This test is not applicable since the TOE mandates the SAN extension for IP address identifiers. See FCS_TLSC_EXT.1.2 test 1.

g) Test 7 [conditional]: If the secure channel is used for FPT_ITT, and RFC 5280 is selected, the evaluator shall perform the following tests. Note, when multiple attribute types are selected in the SFR (e.g. when multiple attribute types are combined to form the unique identifier), the evaluator modifies each attribute type in accordance with the matching criteria described in the TSS (e.g. creating a mismatch of one attribute type at a time while other attribute types contain values that will match a portion of the reference identifier):

1) The evaluator shall present a server certificate that does not contain an identifier in the Subject (DN) attribute type(s) that matches the reference identifier. The evaluator shall verify that the connection fails.

2) The evaluator shall present a server certificate that contains a valid identifier as an attribute type other than the expected attribute type (e.g. if the TOE is configured to expect id-at-serialNumber=correct_identifier, the certificate could instead include id-at-name=correct_identifier), and does not contain the SAN extension. The evaluator shall verify that the connection fails. Remark: Some systems might require the presence of the SAN extension. In this case the connection would still fail but for the reason of the missing SAN extension instead of the mismatch of CN and reference identifier. Both reasons are acceptable to pass this test.

3) The evaluator shall present a server certificate that contains a Subject attribute type that matches the reference identifier and does not contain the SAN extension. The evaluator shall verify that the connection succeeds.

4) The evaluator shall confirm that all use of wildcards results in connection failure regardless of whether the wildcards are used in the left or right side of the presented identifier. (Remark: Use of wildcards is not addressed within RFC 5280.)

Findings:	FPT_ITT with RFC 5280 is not claimed.

FCS_TLSC_EXT.1.3

231 The evaluator shall demonstrate that using an invalid certificate results in the function failing as follows:

232 Test 1: Using the administrative guidance, the evaluator shall load a CA certificate or certificates needed to validate the presented certificate used to authenticate an external entity and demonstrate that the function succeeds, and a trusted channel can be established.

Findings:	This test case is performed in conjunction with FIA_X509_EXT.1.

233 Test 2: The evaluator shall then change the presented certificate(s) so that validation fails and show that the certificate is not automatically accepted. The evaluator shall repeat this test to cover the selected types of failure defined in the SFR (i.e. the selected ones from failed matching of the reference identifier, failed validation of the certificate path, failed validation of the expiration date, failed determination of the revocation status). The evaluator performs the action indicated in the SFR selection observing the TSF resulting in the expected state for the trusted channel (e.g. trusted channel was established) covering the types of failure for which an override mechanism is defined.

Findings:	This test case is performed in conjunction with FIA_X509_EXT.1.

234 Test 3 [conditional]: The purpose of this test to verify that only selected certificate validation failures could be administratively overridden. If any override mechanism is

defined for failed certificate validation, the evaluator shall configure a new presented certificate that does not contain a valid entry in one of the mandatory fields or parameters (e.g. inappropriate value in extendedKeyUsage field) but is otherwise valid and signed by a trusted CA. The evaluator shall confirm that the certificate validation fails (i.e. certificate is rejected), and there is no administrative override available to accept such certificate.

Findings:	This test case is performed in conjunction with FIA_X509_EXT.1.

FCS_TLSC_EXT.1.4

235 Test 1 [conditional]: If the TOE presents the Supported Elliptic Curves/Supported Groups Extension, the evaluator shall configure the server to perform ECDHE or DHE (as applicable) key exchange using each of the TOE's supported curves and/or groups. The evaluator shall verify that the TOE successfully connects to the server.

Findings:	**Pass** The evaluator attempted TLS connections from the TOE using each of the selected supported curves/groups in the [ST] and verified that the connection succeeded.

5.1.3 FCS_TLSS_EXT.1 Extended: TLS Server Protocol

5.1.3.1 TSS

FCS_TLSS_EXT.1.1

236 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 section 6.2.11 of the ST TSS, the TLS server ciphersuites are defined (pointing back to FCS_TLSS_EXT.1.1 in section 5.3.2). These ciphersuites are consistent with the permissible set defined in the SFR. These ciphersuites are consistent with the claimed cryptographic components from FCS_COP.1.

FCS_TLSS_EXT.1.2

237 The evaluator shall verify that the TSS contains a description of how the TOE technically prevents the use of old SSL and TLS versions.

Findings:	ST TSS section 6.2.11 explicitly states that the TOE will reject any protocol version other than TLS 1.2.

FCS_TLSS_EXT.1.3

238 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:	ST Section 6.2.11 points back to Section 5.3.2 FCS_TLSS_EXT.1 which describes the key agreement parameters for ECDHE ciphersuites.

FCS_TLSS_EXT.1.4

239 **(NIAP TD0569 applied)** The evaluator shall verify that the TSS describes if session resumption based on session IDs is supported (RFC 4346 and/or RFC 5246) and/or if session resumption based on session tickets is supported (RFC 5077).

240 If the TOE claims a (D)TLS server capable of session resumption (as a single context, or across multiple contexts), the evaluator verifies that the TSS describes how session resumption operates (i.e. what would trigger a full handshake, e.g. checking session status, checking Session ID, etc.). If multiple contexts are used the TSS describes how session resumption is coordinated across those contexts. In case session establishment and session resumption are always using a separate context, the TSS shall describe how the contexts interact with respect to session resumption (in particular regarding the session ID). It is acceptable for sessions established in one context to be resumable in another context.

Findings:	ST Section 5.3.2 confirms that the TOE does not support session resumption or session tickets.

241 If session tickets are supported, the evaluator shall verify that the TSS describes that the session tickets are encrypted using symmetric algorithms consistent with FCS_COP.1/DataEncryption. The evaluator shall verify that the TSS identifies the key lengths and algorithms used to protect session tickets.

Findings:	ST Section 5.3.2 confirms that the TOE does not support session resumption or session tickets.

242 If session tickets are supported, the evaluator shall verify that the TSS describes that session tickets adhere to the structural format provided in section 4 of RFC 5077 and if not, a justification shall be given of the actual session ticket format.

Findings:	ST Section 5.3.2 confirms that the TOE does not support session resumption or session tickets.

5.1.3.2 Guidance Documentation

FCS_TLSS_EXT.1.1

243 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:	There is no additional configuration required; the TOE conforms to the required configuration by default.

FCS_TLSS_EXT.1.2

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

Findings:	There is no additional configuration required; the TOE conforms to the required configuration by default.

FCS_TLSS_EXT.1.3

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

Findings:	There is no additional configuration required; the TOE conforms to the required configuration by default. To replace the web server certificate, the instructions are described in [SUPP] section 2.7 (or also under section "***Complete a CA certificate signing request (CSR) for the controllers***" in the [REST]).

5.1.3.3 Tests

FCS_TLSS_EXT.1.1

246 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).

Findings:	**Pass** The evaluator successfully attempted TLS connections to the TOE using each of the claimed ciphersuites in the [ST].

247 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.

Findings:	**Pass** The evaluator attempted a TLS connection to the TOE using TLS_RSA_WITH_RC4_128_SHA and verified that the connection was rejected. The evaluator then attempted another connection to the TOE using TLS_NULL_WITH_NULL_NULL and verified that it was also rejected.

248 Test 3: The evaluator shall perform the following modifications to the traffic:

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

Findings:	**Pass** The evaluator attempted a connection to the TOE with a modified byte in the Client Finished message and verified that the connection was rejected.

(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 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'.

Findings:	**Pass** The evaluator successfully attempted a TLS connection to the TOE and verified via a packet capture that the Server Finished message was appropriately encrypted.

FCS_TLSS_EXT.1.2

249 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.

Findings:	**Pass** The evaluator attempted connections to the TOE using each of the protocol versions selected in the [ST] and verified that the connections were rejected.

FCS_TLSS_EXT.1.3

250 Test 1: [conditional] If ECDHE ciphersuites are supported:

a) The evaluator shall repeat this test for each supported elliptic curve. The evaluator shall attempt a connection using a supported ECDHE ciphersuite and a single supported elliptic curve specified in the Elliptic Curves Extension. The Evaluator shall verify (though a packet capture or instrumented client) that the TOE selects the same curve in the Server Key Exchange message and successfully establishes the connection.

Findings:	**Pass**

	The evaluator successfully attempted TLS connections to the TOE using each of the claimed key exchanges in the [ST].

b) The evaluator shall attempt a connection using a supported ECDHE ciphersuite and a single unsupported elliptic curve (e.g. secp192r1 (0x13)) specified in RFC4492, chap. 5.1.1. The evaluator shall verify that the TOE does not send a Server Hello message and the connection is not successfully established.

Findings:	**Pass** The evaluator attempted a connection to the TOE using the secp192r1 curve and verified that the TOE rejected the connection.

251 Test 2: [conditional] If DHE ciphersuites are supported, the evaluator shall repeat the following test for each supported parameter size. If any configuration is necessary, the evaluator shall configure the TOE to use a supported Diffie-Hellman parameter size. The evaluator shall attempt a connection using a supported DHE ciphersuite. The evaluator shall verify (through a packet capture or instrumented client) that the TOE sends a Server Key Exchange Message where p Length is consistent with the message are the ones configured Diffie-Hellman parameter size(s).

Findings:	**Pass** The evaluator successfully attempted TLS connections to the TOE using each of the claimed DHE groups in the [ST].

252 Test 3: [conditional] If RSA key establishment ciphersuites are supported, the evaluator shall repeat this test for each RSA key establishment key size. If any configuration is necessary, the evaluator shall configure the TOE to perform RSA key establishment using a supported key size (e.g. by loading a certificate with the appropriate key size). The evaluator shall attempt a connection using a supported RSA key establishment ciphersuite. The evaluator shall verify (through a packet capture or instrumented client) that the TOE sends a certificate whose modulus is consistent with the configured RSA key size.

Findings:	This test is not applicable because no RSA key establishment ciphersuites are claimed by the [ST].

FCS_TLSS_EXT.1.4

Test Objective: To demonstrate that the TOE will not resume a session for which the client failed to complete the handshake (independent of TOE support for session resumption).

253 **(NIAP TD0569 applied)** Test 1: [conditional] : If the TOE does not support session resumption based on session IDs according to RFC4346 (TLS1.1) or RFC5246 (TLS1.2) or session tickets according to RFC5077, the evaluator shall perform the following test:

a) The client sends a Client Hello with a zero-length session identifier and with a SessionTicket extension containing a zero-length ticket.

b) The client verifies the server does not send a NewSessionTicket handshake message (at any point in the handshake).

c) The client verifies the Server Hello message contains a zero-length session identifier or passes the following steps: Note: The following steps are only performed if the ServerHello message contains a non-zero length SessionID.

d) The client completes the TLS handshake and captures the SessionID from the ServerHello.

e) The client sends a ClientHello containing the SessionID captured in step d). This can be done by keeping the TLS session in step d) open or start a new TLS session using the SessionID captured in step d).

f) The client verifies the TOE (1) implicitly rejects the SessionID by sending a ServerHello containing a different SessionID and by performing a full handshake (as shown in Figure 1 of RFC 4346 or RFC 5246), or (2) terminates the connection in some way that prevents the flow of application data.

Remark: If multiple contexts are supported for session resumption, the session ID or session ticket may be obtained in one context for resumption in another context. It is possible that one or more contexts may only permit the construction of sessions to be reused in other contexts but not actually permit resumption themselves. For contexts which do not permit resumption, the evaluator is required to verify this behaviour subject to the description provided in the TSS. It is not mandated that the session establishment and session resumption share context. For example, it is acceptable for a control channel to establish and application channel to resume the session.

Findings:	**Pass** The evaluator successfully connected to the TOE then attempted another connection reusing the session identifier from the first connected. The evaluator confirmed that the TOE rejected the reuse of the previous session identifier and verified that a new session identifier was issued by the TOE.

254 **(NIAP TD0569 applied)** Test 2: [conditional] : If the TOE supports session resumption using session IDs according to RFC4346 (TLS1.1) or RFC5246 (TLS1.2), the evaluator shall carry out the following steps (note that for each of these tests, it is not necessary to perform the test case for each supported version of TLS):

a) The evaluator shall conduct a successful handshake and capture the TOE-generated session ID in the Server Hello message. The evaluator shall then initiate a new TLS connection and send the previously captured session ID to show that the TOE resumed the previous session by responding with ServerHello containing the same SessionID immediately followed by ChangeCipherSpec and Finished messages (as shown in Figure 2 of RFC 4346 or RFC 5246).

b) The evaluator shall initiate a handshake and capture the TOE-generated session ID in the Server Hello message. The evaluator shall then, within the same handshake, generate or force an unencrypted fatal Alert message immediately before the client would otherwise send its ChangeCipherSpec message thereby disrupting the handshake. The evaluator shall then initiate a new Client Hello using the previously captured session ID, and verify that the server (1) implicitly rejects the session ID by sending a ServerHello containing a different SessionID and performing a full handshake (as shown

in figure 1 of RFC 4346 or RFC 5246), or (2) terminates the connection in some way that prevents the flow of application data.

Remark: If multiple contexts are supported for session resumption, for each of the above test cases, the session ID may be obtained in one context for resumption in another context. There is no requirement that the session ID be obtained and replayed within the same context subject to the description provided in the TSS. All contexts that can reuse a session ID constructed in another context must be tested. It is not mandated that the session establishment and session resumption share context. For example, it is acceptable for a control channel to establish and application channel to resume the session.

Findings:	This test is not applicable because the ST does not claim session resumption using session IDs

255 **(NIAP TD0569 applied)** Test 3: [conditional] : If the TOE supports session tickets according to RFC5077, the evaluator shall carry out the following steps (note that for each of these tests, it is not necessary to perform the test case for each supported version of TLS):

a) [TD0556 and NITDecisionRfI202024]The evaluator shall permit a successful TLS handshake to occur in which a session ticket is exchanged with the non-TOE client. The evaluator shall then attempt to correctly reuse the previous session by sending the session ticket in the ClientHello. The evaluator shall confirm that the TOE responds with an abbreviated handshake described in section 3.1 of RFC 5077 and illustrated with an example in figure 2. Of particular note: if the server successfully verifies the client's ticket, then it may renew the ticket by including a NewSessionTicket handshake message after the ServerHello in the abbreviated handshake (which is shown in figure 2). This is not required, however as further clarified in section 3.3 of RFC 5077.

b) The evaluator shall permit a successful TLS handshake to occur in which a session ticket is exchanged with the non-TOE client. The evaluator will then modify the session ticket and send it as part of a new Client Hello message. The evaluator shall confirm that the TOE either (1) implicitly rejects the session ticket by performing a full handshake (as shown in figure 3 or 4 of RFC 5077), or (2) terminates the connection in some way that prevents the flow of application data.

Remark: If multiple contexts are supported for session resumption, for each of the above test cases, the session ticket may be obtained in one context for resumption in another context. There is no requirement that the session ticket be obtained and replayed within the same context subject to the description provided in the TSS. All contexts that can reuse a session ticket constructed in another context must be tested. It is not mandated that the session establishment and session resumption share context. For example, it is acceptable for a control channel to establish and application channel to resume the session.

Findings:	This test is not applicable because the ST does not claim session tickets.

5.2 Identification and Authentication (FIA)

5.2.1 FIA_X509_EXT.1/Rev X.509 Certificate Validation

5.2.1.1 TSS

256 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:	In the ST TSS section 6.3.6, X.509 certificates are validated at the TLS client and when imported into the TOE. HTTPS is HTTP over TLS and is therefore included in the description implicitly. The description of the validation process includes a summary of the X.509 processing algorithm from RFC 5280 as well as the specific checks the TOE performs to meet the requirements regarding expiry dates, extended key usage, basic constraints and so on. The TOE employs OCSP for certificate revocation validation. The status of X.509 certificates are checked each time they are used with the exception of the TOE's own certificate used to identify itself for incoming HTTPS requests to satisfy the trusted path over HTTP/TLS. For all other uses, the certificates are checked on each use. Regarding the check for extendedKeyUsage OIDs, the TOE will check for the OIDs it supports and expects. If additional OIDs are contained in the certificate, they are not checked. As X.509 certificates are not used for trusted updates, firmware integrity self-tests or client authentication, the code-signing and clientAuthentication purpose is not checked in the extendedKeyUsage for related certificates.

257 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.

Findings:	As per the ST TSS section 6.3.6**,** the TOE performs revocation checking on the server certificate presented to the TOE TLS client during secure TLS transaction with a server, import of the CA certificate to the TOE, installing of TOE server certificate and any device-related certificates. If, during the entire trust chain verification activity, any certificate under review fails a verification check, then the entire trust chain is deemed untrusted.

5.2.1.2 Guidance Documentation

258 The evaluator shall also ensure that the guidance documentation describes where the check of validity of the certificates takes place, describes 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) and describes how certificate revocation checking is performed and on which certificate.

Findings:	The Section ***Enable certificate revocation checking*** of [REST] covers instructions on enabling certification revocation for certificates. The section also describes the revocation process executed by TOE to accept or reject the certificate.

5.2.1.3 Tests

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.

Findings:	**Pass** The successful connection from the TOE for test 1a is performed in conjunction with FCS_TLSC_EXT.1.1 Test 1. The evaluator then removed the root CA from the TOE's trust store and attempted a TLS connection from the TOE and verified that the connection was rejected by the TOE.

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

Findings:	**Pass** The evaluator attempted a TLS connection from the TOE to a server presenting an expired certificate and verified that the connection was rejected by the TOE.

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.

Findings:	**Pass** Note OCSP is only selected in the [ST]. The evaluator attempted a TLS connection from the TOE to a server with a valid certificate to be checked via OCSP. The evaluator confirmed that the TOE appropriately queried the OCSP responder and accepted the connection. The evaluator then repeated this test with a revoked certificate and the TOE rejected the connection. The evaluator then repeated this test using a revoked intermediate CA certificate and verified that the connection was rejected by the TOE.

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.

Findings:	**Pass** The evaluator attempted a TLS connection from the TOE to a server presenting valid certificates to be checked via OCSP. The TOE appropriately queried the OCSP responder however the responder signed the response with a certificate without the OCSP signing purpose. The evaluator confirmed that the TOE rejected the connection.

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.)

Findings:	**Pass** The evaluator attempted a TLS connection to a server with a modified byte in the certificate in the first eight bytes and verified that the TOE rejected the connection.

f) Test 6: The evaluator shall modify any byte in the certificate signatureValue field (see RFC5280 Sec. 4.1.1.3), which is normally the last field in the certificate, and demonstrate that the certificate fails to validate. (The signature on the certificate will not validate.)

Findings:	**Pass** The evaluator attempted a TLS connection to a server with a modified byte in the certificate signatureValue field and verified that the TOE rejected the connection.

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.)

Findings:	**Pass** The evaluator attempted a TLS connection to a server with a modified byte in the certificate's public key and verified that the TOE rejected the connection.

h) [**NIAP TD0527 and NITDecisionRfI202014.pdf applied**] Test 8: (Conditional on support for EC certificates as indicated in FCS_COP.1/SigGen). The evaluator shall conduct the following tests:

Test 8a: (Conditional on TOE ability to process CA certificates presented in certificate message) The test shall be designed in a way such that only the EC root certificate is designated as a trust anchor, and by setting up the trust store in a way that the EC 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 EC root CA certificate in the trust store). The evaluator shall present the TOE with a valid chain of EC certificates (terminating in a trusted CA certificate), where the elliptic curve parameters are specified as a named curve. The evaluator shall confirm that the TOE validates the certificate chain.

Test 8b: (Conditional on TOE ability to process CA certificates presented in certificate message) The test shall be designed in a way such that only the EC root certificate is designated as a trust anchor, and by setting up the trust store in a way that the EC 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 EC root CA certificate in the trust store). The evaluator shall present the TOE with a chain of EC certificates (terminating in a trusted CA certificate), where the intermediate certificate in the certificate chain uses an explicit format version of the Elliptic Curve parameters in the public key information field, and is signed by the trusted EC root CA, but having no other changes. The evaluator shall confirm the TOE treats the certificate as invalid.

Test 8c: The evaluator shall establish a subordinate CA certificate, where the elliptic curve parameters are specified as a named curve, that is signed by a trusted EC root CA. The evaluator shall attempt to load the certificate into the trust store and observe that it is accepted into the TOE's trust store. The evaluator shall then establish a subordinate CA certificate that uses an explicit format version of the elliptic curve parameters, and that is signed by a trusted EC root CA. The evaluator shall attempt to load the certificate into the trust store and observe that it is rejected, and not added to the TOE's trust store.

Findings:	**Pass** The evaluator attempted a TLS connection from the TOE to a server presenting a certificate that is signed by an intermediate CA with non-explicit EC parameters and verified that the connection succeeded. The evaluator then attempted a TLS connection from the TOE to a server presenting a certificate that is signed by an intermediate CA with explicitly defined EC parameters and verified that the TOE rejected the connection. The evaluator then attempted to upload the Intermediate CA certificate with explicitly defined EC parameters to the TOE’s trust store and confirmed the import was denied.

259 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.

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

261 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).

Findings:	**Pass** The evaluator attempted a TLS connection from the TOE to a server with a certificate that chains to a root CA without the basicConstraints extension and verified that the connection was rejected by the TOE.

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).

Findings:	**Pass** The evaluator attempted a TLS connection from the TOE to a server with a certificate that chains to a root CA with the basicConstraints extension set to FALSE and verified that the connection was rejected by the TOE.

262 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:	Note the TOE only uses the TLS client for certificate processing so this is satisfied by the testing above.

5.2.2 FIA_X509_EXT.2 X.509 Certificate Authentication

5.2.2.1 TSS

263 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:	Section 6.3.7 in the ST TSS indicates there is a trust store for certificates to be stored and examined as part of the validation process that is described in section 6.3.6. Section 6.3.7 also provides pointers to the Guidance documents for configuration instructions.

264 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:	The ST TSS in section 6.3.7 claims that the TOE will not permit the connection to succeed if the OCSP responder cannot be contacted. There are no distinctions provided for any specific trusted channel. The administrator can configure the OCSP in two ways: by default, the TOE will extract the OCSP responder URI from the Authority Information Access field or if configured, the TOE will use a single centralized OCSP responder for all revocation checks.

5.2.2.2 Guidance Documentation

The evaluator shall also ensure that the guidance documentation describes the configuration required in the operating environment so the TOE can use the certificates. The guidance documentation shall also include any required configuration on the TOE to use the certificates. The guidance document shall also describe the steps for the Security Administrator to follow if the connection cannot be established during the validity check of a certificate used in establishing a trusted channel.

Findings:	The Section ***Enable certificate revocation checking*** of [REST] covers instructions on enabling certification revocation for certificates. The section also describes the revocation process executed by TOE to accept or reject the certificate. The FAQ part of the Section ***Certificate*** of [REST] describes different scenarios for the trusted connection failure between TOE and other operating environment components and also provides recommended actions.

5.2.2.3 Tests

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

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.

Findings:	**Pass** The evaluator attempted a TLS connection from the TOE to a server presenting a certificate to be checked via OCSP. The evaluator ensured that the OCSP responder was offline and observed that the TOE attempted to reach the OCSP responder and ultimately failed and rejected the TLS connection.

5.2.3 FIA_X509_EXT.3 Extended: X509 Certificate Requests

5.2.3.1 TSS

266 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 ST claims “device specific” information and outlines the information claimed in section 6.3.8 of the ST TSS.

5.2.3.2 Guidance Documentation

267 The evaluator shall check to ensure that the guidance documentation contains instructions on requesting certificates from a CA, including generation of a CertificateRequests. 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:	To replace the web server certificate, the instructions described in [SUPP] section 2.7 (or also under section “***Complete a CA certificate signing request (CSR) for the controllers***” in the [REST]).

5.2.3.3 Tests

268 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 request 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.

Findings:	**Pass** The evaluator generated a certificate request on the TOE and inspected it to confirm that it contained all of the values specified in the [ST].

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 response message, and demonstrate that the function succeeds.

Findings:	**Pass** The evaluator signed the certificate request from test 1 via an untrusted CA and attempted to import it onto the TOE. The evaluator confirmed that this attempt was rejected by the TOE. The evaluator then imported the necessary certificates to validate the signed request onto the TOE then attempted to import it again, this time successfully.

5.3 Security management (FMT)

5.3.1 FMT_MOF.1/Functions Management of security functions behaviour

5.3.1.1 TSS

269 For distributed TOEs see chapter 3.4.1.1.

Findings:	The TOE is not a distributed TOE**.**

270 For non-distributed TOEs, the evaluator shall ensure the TSS for each administrative function identified the TSS details how the Security Administrator determines or modifies the behaviour of (whichever is supported by the TOE) transmitting audit data to an external IT entity, handling of audit data, audit functionality when Local Audit Storage Space is full (whichever is supported by the TOE).

Findings:	The ST section 6.4.2 claims that the TOE restricts the ability to modify (enable/disable) transmission of audit records to an external audit server to Security Administrators as per the selections identified in ST Section 5.3.1 FAU_GEN.1: a) Start-up and shut-down of the audit functions; b) All auditable events for the not specified level of audit; c) Administrative login and logout (name of user account shall be logged if individual user accounts are required for Administrators); d) Changes to TSF data related to configuration changes (in addition to the information that a change occurred it shall be logged what has been changed); e) Generating/import of, changing, or deleting of cryptographic keys (in addition to the action itself a unique key name or key reference shall be logged); f) Resetting passwords (name of related user account shall be logged); and g) Specifically defined auditable events listed in Table 11.

5.3.1.2 Guidance Documentation

271 For distributed TOEs see chapter 3.4.1.2.

Findings:	The TOE is not a distributed TOE.

272 For non-distributed TOEs, the evaluator shall also ensure the Guidance Documentation describes how the Security Administrator determines or modifies the behaviour of (whichever is supported by the TOE) transmitting audit data to an external IT entity, handling of audit data, audit functionality when Local Audit Storage Space is full (whichever is supported by the TOE) are performed to include required configuration settings.

Findings:	Section 2.2 of [SUPP] contains detailed instructions on configuring external secure audit log channel for TOE to transmit local audit data to an external syslog server. Section ***Access Management → Define Audit Log Policies*** of [REST] contains instructions on configuring audit overwrite policy to address logging when the maximum space is reached.

5.3.1.3 Tests

273 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:	**Pass** The evaluator attempted to modify the TOE's syslog settings as a user that does not have the Security Administrator role and confirmed that the changes were denied by the TOE.

274 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.

Findings:	**Pass** The evaluator attempted to modify the TOE's syslog settings as a user that does have the Security Administrator role and confirmed that the changes were accepted by the TOE.

275 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.

276 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 test is not applicable because these selections are not made by the [ST].

277 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.

Findings:	This test is not applicable because these selections are not made by the [ST].

278 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.

279 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 test is not applicable because these selections are not made by the [ST].

280 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.

281 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 test is not applicable because these selections are not made by the [ST].

282 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.

Findings:	This test is not applicable because these selections are not made by the [ST].

283 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 Security Administrator authentication shall be successful.

Findings:	This test is not applicable because these selections are not made by the [ST].

5.3.2 FMT_MTD.1/CryptoKeys Management of TSF Data

5.3.2.1 TSS

284 For distributed TOEs see chapter 3.4.1.1.

Findings:	The TOE is not a distributed TOE.

285 For non-distributed TOEs, the evaluator shall ensure the TSS lists the keys the Security Administrator is able to manage to include the options available (e.g. generating keys, importing keys, modifying keys or deleting keys) and how that how those operations are performed.

Findings:	The ST TSS section 6.4.4 claims that the TOE restricts the ability to modify, delete, generate, import or otherwise manage TLS and any configured X.509 private keys to Security Administrators via web GUI.

5.3.2.2 Guidance Documentation

286 For distributed TOEs see chapter 3.4.1.2.

Findings:	The TOE is not a distributed TOE

287 For non-distributed TOEs, the evaluator shall also ensure the Guidance Documentation lists the keys the Security Administrator is able to manage to include the options available (e.g. generating keys, importing keys, modifying keys or deleting keys) and how that how those operations are performed.

Findings:	The Section ***Certificates*** of [REST] covers concepts and instructions on how to generate a certificate signing request (CSR), complete the signed certificate, and import root/intermediate certificates to the TOE trust store.

5.3.2.3 Tests

288 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.

Findings:	**Pass** The evaluator attempted to modify the certificates on the TOE as a user that does not have the Security Administrator role and confirmed that the changes were rejected by the TOE.

289 The evaluator shall try to perform at least one of the related actions with prior authentication as Security Administrator. This attempt should be successful.

Findings:	**Pass** The evaluator attempted to modify the certificates on the TOE as a user that does have the Security Administrator role and confirmed that the changes were accepted by the TOE.

6 Evaluation Activities for Security Assurance Requirements

6.1 ASE: Security Target

290 When evaluating a Security Target, the evaluator performs the work units as presented in the CEM. In addition, the evaluator ensures the content of the TSS in the ST satisfies the EAs specified in Section 2 (Evaluation Activities for SFRs).

6.2 ADV: Development

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

292 The functional specification describes the TOE Security Functions Interfaces (TSFIs). It is not necessary to have a formal or complete specification of these interfaces.

293 No additional "functional specification" documentation is necessary to satisfy the Evaluation Activities specified in [SD].

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

295 5.2.1.1 Evaluation Activity: 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:	From section 7.2.1 of the NDcPP: "For this cPP, the Evaluation Activities for this family focus on understanding the interfaces presented in the TSS in response to the functional requirements and the interfaces presented in the AGD documentation." The ST and the AGD comprise the functional specification. If the test in [SD] cannot be completed because the ST or the AGD are 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 port and REST API), there were no areas that were deficient.

296 5.2.1.2 Evaluation Activity: 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:	See comments in the previous work unit.

297 5.2.1.3 Evaluation Activity: The evaluator shall examine the interface documentation to develop a mapping of the interfaces to SFRs.

Findings:	See comments in the previous work unit.

6.3 AGD: Guidance

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

299 5.3.1.1 Evaluation Activity: 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:	The operational guidance documentation is distributed with the TOE as built-in on-line 'Help' documentation that is available to administrators anytime they are logged in to the TOE. The documentation is additionally available for download from the NetApp Support web site.

300 5.3.1.2 Evaluation Activity: 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:	There is only one operational environment claimed in the [ST]. All TOE platforms claimed in [ST] are covered by the operational guidance. This is evidenced by the separate Installation guides [SETUP] and the platform equivalency.

301 5.3.1.3 Evaluation Activity: 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 associated with the TOE. No such warning is required since there is no possibility of being able to configure a second engine.

302 5.3.1.4 Evaluation Activity: 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.

1) **Findings:** The [SUPP] document covers configuration of the in-scope functionality where additional configuration might be required. In addition, section 1.3 of the [SUPP]

outlines the in-scope security functionality as well as the interfaces over which these functions are available.

303 5.3.1.5 Evaluation Activity: 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) **(NIAP TD0536 applied)** The documentation must describe the process for verifying updates to the TOE for each method selected for FPT_TUD_EXT.1.3 in the Security Target. The evaluator shall verify that this process includes the following steps:

5) 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).

6) 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:	See work unit [SD] 5.3.1.3 for configuration of the cryptographic engine. The TOE claims published hashes instead of digital signatures. The process for obtaining the update and verifying the published hash is described in [SUPP] section 2.13. The process for manually upgrading the TOE is provided in [SUPP] section 2.8 which includes an indicator as to whether the process was successful or not. See work unit [SD] 5.3.1.4 for details as to what was covered by the EAs.

304 5.3.2.1 Evaluation Activity: The evaluator shall examine the Preparative procedures to ensure they include a description of how the Security 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:	Please refer to work unit AGD_OPE.1-6.

305 5.3.2.2 Evaluation Activity: 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:	There is only one operational environment claimed in the [ST]. All TOE platforms claimed in [ST] are covered by the operational guidance. This is evidenced by the separate Installation guides [SETUP] and the platform equivalency.

306 5.3.2.3 Evaluation Activity: The evaluator shall examine the preparative procedures to ensure they include instructions to successfully install the TSF in each Operational Environment.

Findings:	See previous work unit.

307 5.3.2.4 Evaluation Activity: 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], [SETUP] and [REST] documentation provide extensive information on managing the security of the TOE as an individual product. Additional best practice guidance provided within those documents help instill a culture of secure manageability within a larger operational environment.

308 5.3.2.5 Evaluation Activity: In addition, the evaluator shall ensure that the following requirements are also met.

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 does not have default TOE passwords. When the TOE is configured for the first time the admin password must be set. The other four built-in accounts (storage, security, support and monitor) have no password associated with them, but they cannot be used until a password is assigned.

7 Vulnerability Assessment

309 5.6.1.1 Evaluation Activity: 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.

310 **(NIAP TD0547 applied)** The developer shall provide documentation identifying the list of software and hardware components 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 of the TOE), for example a web server, protocol or cryptographic libraries, (independently identifiable and reusable components are not limited to the list provided in the example). 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 vulnerability hypotheses during their analysis.

Findings:	The developer provided a list of software that composed the TOE. The evaluator also added 'NetApp E-Series' and 'NetApp EF-Series' to this list. This list was used to perform a public survey for potential vulnerabilities.

311 5.6.1.2 Evaluation Activity: 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 evaluator generated flaw hypotheses and generated a report including a vulnerability analysis and results as specified in Appendix A.