

Ixia, A Keysight Business

Vision Series Network Packet Broker v5.3.0

Assurance Activity Report

Version 0.9

July 2020

Document prepared by



www.lightshipsec.com

Table of Contents

1	INTRODUCTION.....	3
1.1	EVALUATION IDENTIFIERS	3
1.2	EVALUATION METHODS.....	3
2	TOE DETAILS.....	7
2.1	OVERVIEW	7
2.2	TOE MODELS.....	7
2.3	REFERENCE DOCUMENTS	7
2.4	SUMMARY OF SFRS	8
3	EVALUATION ACTIVITIES FOR SFRS.....	10
3.1	SECURITY AUDIT (FAU).....	10
3.2	CRYPTOGRAPHIC SUPPORT (FCS).....	16
3.3	IDENTIFICATION AND AUTHENTICATION (FIA).....	32
3.4	SECURITY MANAGEMENT (FMT).....	37
3.5	PROTECTION OF THE TSF (FPT).....	41
3.6	TOE ACCESS (FTA).....	49
3.7	TRUSTED PATH/CHANNELS (FTP).....	53
4	EVALUATION ACTIVITIES FOR SELECTION-BASED REQUIREMENTS.....	57
4.1	CRYPTOGRAPHIC SUPPORT (FCS).....	57
4.2	IDENTIFICATION AND AUTHENTICATION (FIA).....	71
4.3	SECURITY MANAGEMENT (FMT).....	77

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	Ixia, A Keysight Business
TOE	Vision Series Network Packet Broker v5.3.0
Security Target	Ixia, A Keysight Business Vision Series Network Packet Broker v5.3.0 Security Target, v1.4
Protection Profile	collaborative Protection Profile for Network Devices, v2.1 (NDcPP)

1.2 Evaluation Methods

2 The evaluation was performed using the methods, tools 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, v2.1 (NDcPP-SD)
Interpretations	NDcPP v2.1
	TD0395: NIT Technical Decision for Different Handling of TLS1.1 and TLS1.2 <i>This TD does not apply to the TOE because it does not claim FCS_TLSS_EXT.2.</i>
	TD0396: NIT Technical Decision for FCS_TLSC_EXT.1.1, Test 2 <i>This TD applies to the TOE.</i>
	TD0397: NIT Technical Decision for Fixing AES-CTR Mode Tests <i>This TD applies to the TOE.</i>

	<p>TD0398: NIT Technical Decision for FCS_SSH*EXT.1.1 RFCs for AES-CTR</p> <p><i>This TD does not apply to the TOE because SSH is not claimed.</i></p>
	<p>TD0399: NIT Technical Decision for Manual installation of CRL (FIA_X509_EXT.2)</p> <p><i>This TD applies to the TOE.</i></p>
	<p>TD0400: NIT Technical Decision for FCS_CKM.2 and elliptic curve-based key establishment</p> <p><i>This TD applies to the TOE.</i></p>
	<p>TD0401: NIT Technical Decision for Reliance on external servers to meet SFRs</p> <p><i>This TD applies to the TOE.</i></p>
	<p>TD0402: NIT Technical Decision for RSA-based FCS_CKM.2 Selection</p> <p><i>This TD applies to the TOE.</i></p>
	<p>TD0407: NIT Technical Decision for handling Certification of Cloud Deployments</p> <p><i>This TD does not apply to the TOE since TOE is not a cloud deployment.</i></p>
	<p>TD0408: NIT Technical Decision for local vs. remote administrator accounts</p> <p><i>This TD applies to the TOE</i></p>
	<p>TD0409: NIT decision for Applicability of FIA_AFL.1 to key-based SSH authentication</p> <p><i>This TD does not apply to the TOE because SSH is not claimed.</i></p>
	<p>TD0410: NIT technical decision for Redundant assurance activities associated with FAU_GEN.1</p> <p><i>This TD applies to the TOE</i></p>
	<p>TD0411: NIT Technical Decision for FCS_SSHC_EXT.1.5, Test 1 - Server and client side seem to be confused</p> <p><i>This TD does not apply to the TOE because SSH is not claimed.</i></p>
	<p>TD0412: NIT Technical Decision for FCS_SSHS_EXT.1.5 SFR and AA discrepancy</p> <p><i>This TD does not apply to the TOE because SSH is not claimed.</i></p>
<p>TD0423: NIT Technical Decision for Clarification about application of RfI#201726rev2</p> <p><i>This TD applies to the TOE</i></p>	

	<p>TD0424: NIT Technical Decision for NDcPP v2.1 Clarification - FCS_SSHC/S_EXT1.5</p> <p><i>This TD does not apply to the TOE because SSH is not claimed.</i></p>
	<p>TD0425: NIT Technical Decision for Cut-and-paste Error for Guidance AA</p> <p><i>This TD applies to the TOE</i></p>
	<p>TD0447: NIT Technical Decision for Using 'diffie-hellman-group-exchange-sha256' in FCS_SSHC/S_EXT.1.7</p> <p><i>This TD does not apply to the TOE because SSH is not claimed.</i></p>
	<p>TD0450: NIT Technical Decision for RSA-based ciphers and the Server Key Exchange message</p> <p><i>This TD applies to the TOE</i></p>
	<p>TD0451: NIT Technical Decision for ITT Comm UUID Reference Identifier</p> <p><i>This TD applies to the TOE.</i></p>
	<p>TD0453: NIT Technical Decision for Clarify authentication methods SSH clients can use to authenticate SSH se</p> <p><i>This TD does not apply to the TOE because the TOE does not include an SSH client.</i></p>
	<p>TD0475: NIT Technical Decision for Separate traffic consideration for SSH rekey</p> <p><i>This TD does not apply to the TOE because the TOE does not include an SSH client or SSH server.</i></p>
	<p>TD0477: NIT Technical Decision for Clarifying FPT_TUD_EXT.1 Trusted Update</p> <p><i>This TD applies to the TOE</i></p>
	<p>TD0478: NIT Technical Decision for Application Notes for FIA_X509_EXT.1 iterations</p> <p><i>This TD applies to the TOE</i></p>
	<p>TD0480: NIT Technical Decision for Granularity of audit events</p> <p><i>This TD applies to the TOE</i></p>
	<p>TD0481: NIT Technical Decision for FCS_(D)TLSC_EXT.X.2 IP addresses in reference identifiers</p> <p><i>This TD applies to the TOE</i></p>
<p>TD0482: NIT Technical Decision for Identification of usage of cryptographic schemes</p> <p><i>This TD applies to the TOE</i></p>	

	<p>TD0483: NIT Technical Decision for Applicability of FPT_APW_EXT.1</p> <p><i>This TD applies to the TOE</i></p>	
	<p>TD0484: NIT Technical Decision for Interactive sessions in FTA_SSL_EXT.1 & FTA_SSL.3</p> <p><i>This TD applies to the TOE</i></p>	

2 TOE Details

2.1 Overview

- 1 This Security Target (ST) defines the Ixia, A Keysight Business Vision Series Network Packet Broker v5.3.0 Target of Evaluation (TOE) for the purposes of Common Criteria (CC) evaluation.
- 2 Ixia Network Packet Brokers are at the heart of all Ixia Visibility solutions. Their main function is to aggregate, load balance and filter network traffic before it's processed by network security and performance monitoring tools.

2.2 TOE models

Model	CPU
Vision ONE	Intel Core i7-3555LE
Vision 7300/7303	Intel Core i7-3555LE
Vision E40	Atom C2538
Vision E100	Xeon D-1518
Vision E10S	Celeron 3965U
Vision X	Xeon D-1527
TradeVision	Intel Core i7-3555LE

2.3 Reference Documents

Table 3: List of Reference Documents

Ref	Document
[ST]	Ixia, A Keysight Business Vision Series Network Packet Broker v5.3.0 Security Target, v1.4
[AGD]	Ixia, A Keysight Business Vision Series Network Packet Broker v5.3.0 Common Criteria Guide, v1.0 User Guides: Ixia Vision X User Guide v5.3.0, 913-2547-01 Rev B Ixia Vision 7300/7303 User Guide v5.3.0, 913-2548-01 Rev B Ixia Vision ONE User Guide v5.3.0, 913-2549-01 Rev B Ixia Vision Edge 40/100 User Guide v5.3.0, 913-2550-01 Rev B Ixia Vision Edge E10S User Guide v5.3.0, 913-2552-01 Rev B Ixia TradeVision User Guide v5.3.0, 913-2565-01 Rev B

2.4 Summary 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_NTP_EXT.1	NTP Protocol
FCS_RBG_EXT.1	Random Bit Generation
FCS_TLSC_EXT.2	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

Requirement	Title
FMT_MTD.1/CryptoKeys	Management of TSF Data
FMT_SMF.1	Specification of Management Functions
FMT_SMR.2	Restrictions on Security Roles
FPT_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
FPT_TUD_EXT.1	Extended: 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

Table 4: List of SFRs

3 Evaluation Activities for SFRs

3.1 Security Audit (FAU)

3.1.1 FAU_GEN.1 Audit data generation

3.1.1.1 TSS

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

Findings:

This information was found in the ST TSS in section 6.1.1. Actions which can affect private cryptographic keys include generating a CSR (which implicitly includes a private key).

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

Findings:

The TOE is not a distributed TOE.

3.1.1.2 Guidance Documentation

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

Findings:

Annex A of Common Criteria Guide in [AGD] shows an example of each auditable event required by FAU_GEN.1 (including each mandatory or selection-based SFR in the ST, if there is an auditable event associated with the SFR).

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

associated with ensuring that the corresponding guidance documentation satisfies the requirements related to it.

Findings:

The evaluator made his determination of the administrative actions, by using FMT SFRs, FIA_PMG_EXT.1, FPT SFRs, FTA SFRs, as a guide, navigate through the user guide document for administrative actions related to those functional requirements, specifically, configuration changes related to TSF data and TSF. This includes the following sections in the user guide:

- Connecting using the Craft Port Interface
- Common Criteria Configuration Guide
- Viewing and Changing System Settings
- The System menu allows
- Managing Users
- Authentication, Authorization, and Accounting (AAA)
- About Local Syslog Viewer
- Syslog Support
- Software Upgrade/Downgrade and Cold Spare Upgrade Procedures
- NPB Syslog Messages

The evaluator also performed the administrative operations, involving changes to TSF data and TSF behaviour, as part of NDcPP v2.1 Test assurance activities, and confirmed that the documents provide sufficient information and instructions to enable evaluator to conduct NDcPP test requirements successfully.

3.1.1.3 Tests

- 7 The evaluator shall test the TOE's ability to correctly generate audit records by having the TOE generate audit records for the events listed in the table of audit events and administrative actions listed above. This should include all instances of an event: for instance, if there are several different I&A mechanisms for a system, the FIA_UIA_EXT.1 events must be generated for each mechanism. The evaluator shall test that audit records are generated for the establishment and termination of a channel for each of the cryptographic protocols contained in the ST. If HTTPS is implemented, the test demonstrating the establishment and termination of a TLS session can be combined with the test for an HTTPS session. When verifying the test results, the evaluator shall ensure the audit records generated during testing match the format specified in the guidance documentation, and that the fields in each audit record have the proper entries.
- 8 For distributed TOEs the evaluator shall perform tests on all TOE components according to the mapping of auditable events to TOE components in the Security Target. For all events involving more than one TOE component when an audit event is triggered, the evaluator has to check that the event has been audited on both sides (e.g. failure of building up a secure communication channel between the two components). This is not limited to error cases but includes also events about successful actions like successful build up/tear down of a secure communication channel between TOE components.
- 9 Note that the testing here can be accomplished in conjunction with the testing of the security mechanisms directly.

Findings:

These tests are conducted throughout the test plan. Meanwhile, the TOE is not a distributed TOE.

3.1.2 FAU_GEN.2 User identity association

3.1.2.1 TSS

10 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

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

12 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

13 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:

This information was found in section 6.1.3 of the ST and states that audit data is transferred to a Syslog server. Log events are sent in real-time over TLS.

14 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 ST describes that the max amount of audit data can be stored locally configured by an administrator. When local audit data store becomes full the oldest audit record is deleted. Only authorized administrators may view audit records and no capability to modify the audit records is provided.

15 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 ST describes that audit records can be stored locally in the audit store or audit data is transferred to a Syslog server via TLS. Not a distributed TOE.

- 16 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:

This information was found in section 6.1.3 of the ST. When the local audit data store is full, the TOE will delete the oldest audit record.

- 17 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:

This information was found in section 6.1.3 of the ST and states the log events are sent in real time.

- 18 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:

N/A – TOE is not a distributed TOE.

- 19 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:

N/A – TOE is not a distributed TOE.

3.1.3.2 Guidance Documentation

- 20 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:

The evaluator checked the user guide and determined that the Section of "Syslog Support" describes audit server protocol (i.e. syslog), configuration of TLS protection, generation CSR (as it supports mutual authentication), etc. Specifically, it says that "syslog messages are created and sent to each external syslog server configured whenever a new configuration or state changes occur on the system."

- 21 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:

The evaluator checked the user guide and determined that the Section of "How local syslog files work - appending and overwriting files" describes the relationship between the local audit data and audit data that are sent to the audit log server. Specifically, it says that "syslog messages are created and sent to each external syslog server configured whenever a new configuration or state changes occur on the system."

- 22 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 evaluator checked the user guide and determined that the behaviour related to FAU_STG_EXT.1.3 is not configuration, and it is consistent with ST.

3.1.3.3 Tests

- 23 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:

Verification that the data is encrypted is satisfied by FTP_ITC.1 for the logging channel. The logging server is a syslog-ng v3.19.1. The evaluator also verified that the audit data were successfully received by the audit server, and this was done automatically without administrator intervention.

- b) Test 2: The evaluator shall perform operations that generate audit data and verify that this data is stored locally. The evaluator shall perform operations that generate audit data until the local storage space is exceeded and verifies that the TOE complies with the behaviour defined in FAU_STG_EXT.1.3. Depending on the configuration this means that the evaluator has to check the content of the audit data when the audit data is just filled to the maximum and then verifies that
- 1) The audit data remains unchanged with every new auditable event that should be tracked but that the audit data is recorded again after the local storage for audit data is cleared (for the option 'drop new audit data' in FAU_STG_EXT.1.3).
 - 2) The existing audit data is overwritten with every new auditable event that should be tracked according to the specified rule (for the option 'overwrite previous audit records' in FAU_STG_EXT.1.3)
 - 3) The TOE behaves as specified (for the option 'other action' in FAU_STG_EXT.1.3).

High-Level Test Description
<p>The evaluator generated auditable events and verified that they were recorded.</p> <p>The evaluator further exported the audit records to a .csv file.</p> <p>Then the evaluator ran a custom bash command to generate volume of audit data.</p> <p>The evaluator again exported the audit records to a .csv file and compared it with previously saved .csv file. By comparison of two .csv files, the evaluator determined that new auditable events were recorded while oldest audit records were overwritten.</p>
Findings: PASS

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

Findings:
The TOE does not claim this 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

- 24 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:

Section 6.2.1 of the ST shows the key generation sizes for all claimed key generation schemes.

The TSS specifies the use for each of the key generation schemes. RSA key generation is used in the generation of certificate signing requests and ECC key generation is used in when establishing ephemeral elliptical curve Diffie Hellman. This is consistent with the selections and TSS for FCS_TLSC_EXT.2 and FCS_TLSS_EXT.1.

3.2.1.2 Guidance Documentation

- 25 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:

The evaluator verified that the user guide instructs the administrator to generate CSR for web server and syslog client, in “Configure Server Certificate for Web API Communication” and “Adding or Modifying External Syslog Servers”, which include RSA public/private key pair generation. However, it does not allow the administrator to configure RSA key size as there is only one choice, 2048-bit, as per FCS_CKM.1; as for ECDHE, the TOE doesn’t provide administrator option to configure elliptic curves.

3.2.1.3 Tests

- 26 Note: The following tests require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on factory products. Generation of long-term cryptographic keys (i.e. keys that are not ephemeral keys/session keys) might be performed automatically (e.g. during initial start-up). Testing of key generation must cover not only administrator invoked key generation but also automated key generation (if supported).

Key Generation for FIPS PUB 186-4 RSA Schemes

- 27 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 .

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

a. Random Primes:

- Provable primes
- Probable primes

b. Primes with Conditions:

- Primes p_1 , p_2 , q_1, q_2 , p and q shall all be provable primes
- Primes p_1 , p_2 , q_1 , and q_2 shall be provable primes and p and q shall be probable primes
- Primes p_1 , p_2 , q_1, q_2 , p and q shall all be probable primes

29 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

30 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

31 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)

32 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 .

33 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

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

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

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

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

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

37 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.

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

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

39 for each FFC parameter set and key pair.

Diffie-Hellman Group 14

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

Findings:

To demonstrate that all cryptographic requirements are satisfied, the Assurance Activity Report must clearly indicate all SFRs for which a CAVP certificate is claimed and include, at a minimum, the cryptographic operation, the NIST standard, the SFR supported, the CAVP algorithm list name (e.g. AES, KAS, CVL, etc.) and the CAVP Certificate number. The CCTL will verify that the claimed NIST validation complies with the NIAP-approved PP requirements the TOE claims to satisfy. The CCTL verification of the NIST validation will constitute performance of the associated assurance activity.

Module	Services	Algorithms	Certificate
BouncyCastle	TLS/HTTPS X.509v3 Key Encryption	RSA KeyGen (186-4) RSA SigGen (186-4) RSA SigVer (186-4) ECDSA KeyGen (186-4) ECDSA SigGen (186-4) ECDSA SigVer (186-4)	C1551 RSA 3118 ECDSA 1590

3.2.2 FCS_CKM.2 Cryptographic Key Establishment

3.2.2.1 TSS

41 [Modified by TD0482 and TD0449] 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.

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

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

<i>Scheme</i>	<i>SFR</i>	<i>Service</i>
<i>RSA</i>	<i>FCS_TLSS_EXT.1</i>	<i>Administration</i>
<i>ECDH</i>	<i>FCS_SSHC_EXT.1</i>	<i>Audit Server</i>
<i>Diffie-Hellman (Group 14)</i>	<i>FCS_SSHC_EXT.1</i>	<i>Backup Server</i>
<i>ECDH</i>	<i>FCS_IPSEC_EXT.1</i>	<i>Authentication Server</i>

44 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:

Section 6.2.2 of the ST illustrate the various key establishment schemes and the usage for each scheme. Section 6.2.1 of the ST matches the schemes in section 6.2.2 and the FCS_CKM.1 selection.

The information provided in Section 6.2.2 of the ST is sufficient to identify the service associated with the key establishment scheme.

3.2.2.2 Guidance Documentation

45 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 evaluator checked the user guide and verified that the user guide doesn't provide the administrator option of configuration of key establishment schemes, as there is only one scheme supported by the TOE, i.e. ECDHE.

3.2.2.3 Tests

Key Establishment Schemes

46 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

47 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

48 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.

- 49 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.
- 50 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.
- 51 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.
- 52 If key confirmation is supported, the TSF shall perform the above for each implemented approved MAC algorithm.

Validity Test

- 53 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.
- 54 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).
- 55 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 schemes

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

Diffie-Hellman Group 14

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

Findings:

To demonstrate that all cryptographic requirements are satisfied, the Assurance Activity Report must clearly indicate all SFRs for which a CAVP certificate is claimed and include, at a minimum, the cryptographic operation, the NIST standard, the SFR supported, the CAVP algorithm list name (e.g. AES, KAS, CVL, etc.) and the CAVP Certificate number. The CCTL will verify that the claimed NIST validation complies with the NIAP-approved PP requirements the TOE claims to satisfy. The CCTL verification of the NIST validation will constitute performance of the associated assurance activity.

Module	Services	Algorithms	Certificate
BouncyCastle	TLS/HTTPS X.509v3 Key Encryption	KAS-ECC	C1551 KAS 210

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 for¹). 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 include their origin and their storage location.

Keys live in both persistent Flash as well as in RAM. RAM-based keys are plaintext and a handful of keys are – as indicated – encrypted in the Flash memory.

Table 14 describes all relevant keys. The TOE claims cryptographic channels covering TLS for trusted channels. The TOE would be required to persistently store private keys and X.509 public key certificates when acting as a server/peer in TLS capacities which is consistent with the given table. The various session keys are consistent with the protocols.

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:

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

The mechanism by which the TOE destroys plaintext keys in non-volatile memory is described in section 6.5.1 of the ST. The NTP Key is the only key stored in non-volatile memory as plaintext. Destroying the keys can be performed by overwriting with new value of key on the TOE.

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 that for non-volatile memory, zeroization occurs via an invocation of an interface. The Linux file system is used to overwrite the NTP key.

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:

All relevant keys are described in table 14 in section 6.5.1 which include their origin, their storage location and the encryption method used.

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 command² and garbage collection to destroy these persistent copies upon their deletion (this would be explained in TSS and Operational Guidance).

Findings:

This is not applicable as it does not state any situations where key destruction may be delayed or prevented.

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

3.2.4.1 Tests

AES-CBC Known Answer Tests

66 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.

67 **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.

68 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.

69 **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.

70 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.

71 **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

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).

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 $[1, N]$.

72 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.

73 **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 $[1, 128]$.

74 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

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

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

AES-CBC Monte Carlo Tests

77 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]
```

78 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.

79 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

80 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.

81 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.

82 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.

83 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.

[Modified by TD0397]

AES-CTR Known Answer Tests

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

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

the validator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

- 86 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.
- 87 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.
- 88 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].
- 89 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 key sizes 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

- 90 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

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

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

- 92 The ciphertext computed in the 1000th iteration is the result for that trial. This result shall be compared to the result of running 1000 iterations with the same values using a known good implementation. The evaluator shall perform this test using each selected keysize.

Findings:

To demonstrate that all cryptographic requirements are satisfied, the Assurance Activity Report must clearly indicate all SFRs for which a CAVP certificate is claimed and include, at a minimum, the cryptographic operation, the NIST standard, the SFR supported, the CAVP algorithm list name (e.g. AES, KAS, CVL, etc.) and the CAVP Certificate number. The CCTL will verify that the claimed NIST

validation complies with the NIAP-approved PP requirements the TOE claims to satisfy. The CCTL verification of the NIST validation will constitute performance of the associated assurance activity.

Module	Services	Algorithms	Certificate
BouncyCastle	TLS/HTTPS X.509v3 Key Encryption	AES-CBC AES-GCM	C1551 AES 5940

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

3.2.5.1 Tests

ECDSA Algorithm Tests

ECDSA FIPS 186-4 Signature Generation Test

93 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

94 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

95 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.

96 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

97 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.

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

Findings:

To demonstrate that all cryptographic requirements are satisfied, the Assurance Activity Report must clearly indicate all SFRs for which a CAVP certificate is claimed and include, at a minimum, the cryptographic operation, the NIST standard, the SFR supported, the CAVP algorithm list name (e.g. AES, KAS, CVL, etc.) and the CAVP Certificate number. The CCTL will verify that the claimed NIST validation complies with the NIAP-approved PP requirements the TOE claims to satisfy. The CCTL verification of the NIST validation will constitute performance of the associated assurance activity.

Module	Services	Algorithms	Certificate
BouncyCastle	TLS/HTTPS X.509v3 Key Encryption	RSA KeyGen (186-4) RSA SigGen (186-4) RSA SigVer (186-4) ECDSA KeyGen (186-4) ECDSA SigGen (186-4) ECDSA SigVer (186-4)	C1551 RSA 3118 ECDSA 1590 Component 2169

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

3.2.6.1 TSS

99 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:

Section 6.2.6 of the [ST] for FCS_COP.1/Hash indicates how SHA is used for each TOE need for cryptographic hashing services.

3.2.6.2 Guidance Documentation

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

Findings:

The evaluator checked the user guide and determined that it doesn't provide the administrator option of configuring hash sizes as hash size is determined in TLS handshaking process automatically.

3.2.6.3 Tests

101 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.

102 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

103 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

104 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

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

Selected Long Messages Test - Byte-oriented Mode

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

Pseudorandomly Generated Messages Test

107 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:

To demonstrate that all cryptographic requirements are satisfied, the Assurance Activity Report must clearly indicate all SFRs for which a CAVP certificate is claimed and include, at a minimum, the cryptographic operation, the NIST standard, the SFR supported, the CAVP algorithm list name (e.g. AES, KAS, CVL, etc.) and the CAVP Certificate number. The CCTL will verify that the claimed NIST validation complies with the NIAP-approved PP requirements the TOE claims to satisfy. The CCTL verification of the NIST validation will constitute performance of the associated assurance activity.

Module	Services	Algorithms	Certificate
BouncyCastle	TLS/HTTPS X.509v3	SHA-1, SHA-256	C1551 SHS 4693

Module	Services	Algorithms	Certificate
	Key Encryption		
OpenSSL	NTP	SHA-1	C1550 SHS 4692

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

3.2.7.1 TSS

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

Findings:

The key length, block size, hash function and output MAC length are found in table 13 of the ST.

3.2.7.2 Tests

109 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:

To demonstrate that all cryptographic requirements are satisfied, the Assurance Activity Report must clearly indicate all SFRs for which a CAVP certificate is claimed and include, at a minimum, the cryptographic operation, the NIST standard, the SFR supported, the CAVP algorithm list name (e.g. AES, KAS, CVL, etc.) and the CAVP Certificate number. The CCTL will verify that the claimed NIST validation complies with the NIAP-approved PP requirements the TOE claims to satisfy. The CCTL verification of the NIST validation will constitute performance of the associated assurance activity.

Module	Services	Algorithms	Certificate
BouncyCastle	TLS/HTTPS X.509v3 Key Encryption	HMAC-SHA-1, HMAC-SHA-256	C1551 HMAC 3915

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

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

3.2.8.1 TSS

111 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 section 6.2.10 in the ST. Details about the seeding mechanism, assumed min-entropy and noise sources are provided in section 6.2.11 of the ST.

3.2.8.2 Guidance Documentation

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

Findings:

The evaluator checked the user guide and determined that it only provides instructions on enabling FIPS mode, in section of "Enable Server FIPS Encryption", implicitly configuring use of FIPS approved RBG function.

3.2.8.3 Tests

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

114 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).

115 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.

116 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 \leq 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:

To demonstrate that all cryptographic requirements are satisfied, the Assurance Activity Report must clearly indicate all SFRs for which a CAVP certificate is claimed and include, at a minimum, the cryptographic operation, the NIST standard, the SFR supported, the CAVP algorithm list name (e.g. AES, KAS, CVL, etc.) and the CAVP Certificate number. The CCTL will verify that the claimed NIST validation complies with the NIAP-approved PP requirements the TOE claims to satisfy. The CCTL verification of the NIST validation will constitute performance of the associated assurance activity.

Module	Services	Algorithms	Certificate
BouncyCastle	TLS/HTTPS X.509v3 Key Encryption	Hash DRBG	C1551 DRBG 2493

3.3 Identification and Authentication (FIA)

3.3.1 FIA_AFL.1 Authentication Failure Management

3.3.1.1 TSS

117 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 the information 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 a Security Administrator unlocks the account.

118 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 ST describes that local console is not subjected to the lock out mechanisms in section 6.3.5 of the TSS.

3.3.1.2 Guidance Documentation

119 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:

The evaluator checked the user guide and determined that the requirement is covered in section of "Locking Regular Users".

120 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:

The evaluator checked the user guide and determined that the requirement is covered in section of "Recovering when no user can authenticate".

3.3.1.3 Tests

121 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.
- b. Test 2: After reaching the limit for unsuccessful authentication attempts as in Test 1 above, the evaluator shall proceed as follows.

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

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

High-Level Test Description

The evaluator followed user guide to set the threshold to 3 via Web UI.
The evaluator then made login attempts with incorrect credentials via Web UI and REST API. After the threshold was reached, the login with valid credential failed.
The evaluator then logged in to TOE via Web UI using another account as administrator and unlocked the account which was just frozen due to 3 consecutive failed login attempts.
After unlocking the frozen account, the evaluator was able to log in to TOE via Web UI and REST API, using such account, with valid credential.

Findings: PASS

3.3.2 FIA_PMG_EXT.1 Password Management

3.3.2.1 Guidance Documentation

- 122 The evaluator shall examine the guidance documentation to determine that it:
- identifies the characters that may be used in passwords and provides guidance to security administrators on the composition of strong passwords, and
 - provides instructions on setting the minimum password length and describes the valid minimum password lengths supported.

Findings:

The evaluator checked the user guide and determined that this requirement is covered in the Section of "Enabling DoD Security Policies"

3.3.2.2 Tests

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

High-Level Test Description

Set the minimum password length to 15 characters. Attempt to set a password less than the minimum length and show it is not accepted. Attempt to set passwords that fail to include characters from the out-of-the-box password complexity requirements and show they are not accepted. Attempt to set a password that meets the complexity and length requirements and show it is accepted. Show the password can be used on applicable management interfaces to log in successfully.

Show that an admin with privileges can change another user's password and that the audit log reflects this capability.

Findings: PASS

3.3.3 FIA_UIA_EXT.1 User Identification and Authentication

3.3.3.1 TSS

- 124 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:

The login process is described in section 6.3 of the ST TSS. It describes the process uniformly for all defined administrative interfaces (local/serial, remote/web). This description includes username and passwords for all interfaces.

125 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:

These actions are provided in section 6.3.2 of the ST. The TOE claims displaying a TOE banner, viewing the TOE version number via the GUI, access help documents and view alarm notifications.

126 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.

127 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

128 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:

The evaluator checked the user guide and Common Criteria Guide, and verified that it only supports password based authentication for administrators to access the web UI and local console (see "Administration Interfaces" of Common Criteria Guide; it provides instructions on how to access the TOE for initially installation and configuration using default credentials (section of "Government Security Configuration Guide"), and how to create a user and provision the user account with password (section of "Add Users"); and there is not any configuration to limit services provided before login.

3.3.3.3 Tests

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

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

High-Level Test Description
For each of the identified interfaces, do: Log into the identified management interface using a known-good credential and logout. Login into the identified management interface using a known-bad credential and logout. Ensure the appropriate audit messages appear.
Findings: PASS

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

High-Level Test Description
The evaluator went to home page of the TOE management UI (the only remote management interface) and it provided services, which are the same as specified in the ST, prior to I&A, However, those services are not configurable.
Findings: PASS

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

High-Level Test Description
The device does not have any services configured prior to I&A aside from a TOE banner.
Findings: PASS

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

Findings:
The TOE is not a distributed TOE

3.3.4 FIA_UAU_EXT.2 Password-based Authentication Mechanism

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

3.3.5 FIA_UAU.7 Protected Authentication Feedback

3.3.5.1 Tests

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

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

High-Level Test Description
Log into the local management interface. Ensure the password field does not echo plaintext characters as claimed by the ST.
Findings: PASS

3.4 Security management (FMT)

3.4.1 General requirements for distributed TOEs

3.4.1.1 TSS

132 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

133 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

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

Findings:
The TOE is not a distributed TOE.

3.4.2 FMT_MOF.1/ManualUpdate

3.4.2.1 TSS

135 For distributed TOEs see chapter 4.4.1.1. There are no specific requirements for non-distributed TOEs.

Findings:

The TOE is not a distributed TOE.

3.4.2.2 Guidance Documentation

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

Findings:

The evaluator checked that the TOE only supports manual update and it is described in section of "Software Upgrade/Downgrade and Cold Spare Upgrade Procedures".

137 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

138 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.

139 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.

High-Level Test Description

Log into the Web GUI using a non-admin account. No interface is provided for update.

Log into the Web GUI using an admin account, and it was able to perform update using a legitimate update image.

Findings: PASS

3.4.3 FMT_MTD.1/CoreData Management of TSF Data

3.4.3.1 TSS

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

Findings:

According to the described functionality in section 5.3.3 and 6.3.2 SFR FIA_UIA_EXT.1.

The evaluator examined section 6.4.4 of TSS and confirmed that access to the TSF data is restricted to system administrators only.

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

Findings:

Section 6.4.3 & 6.4.5 of the ST describes that access to TSF data and functions such as "generation, importation, or deletion of cryptographic keys" are restricted to security administrators. No TOE functionality is available before identification and authentication occurs.

3.4.3.2 Guidance Documentation

142 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 checked the user guide and was able to identify each of TSF-data-manipulating functions, such as:

- Secure web access
- Local audit logs
- Syslog with TLS protection
- User management
- Reliable time stamp
- Session management
- Login banner
- Local authentication
- Certificate management (including trust store / CA certs configuration)
- Enabling FIPS
- Software upgrade
- Self-testing

143 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:

The evaluator checked the user guide and determined that the requirement is covered in the Section of "Syslog Support"

3.4.4 FMT_SMF.1 Specification of Management Functions

144 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) [Modified by TD0408]

145 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:

Section 6.4.6 of the ST indicates that the TOE may be managed via the CLI (console), GUI (HTTPS) or API (HTTPS) and describes the specific management capabilities.

Section 6.4.6 further specifies which functions can be performed on specific interfaces.

146 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

147 See section 4.4.4.1.

3.4.4.3 Tests

148 The evaluator tests management functions as part of testing the SFRs identified in section 4.4.4. 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.

Note:

There are no explicit test activities and therefore none are recorded here. All management functions in FMT_SMF.1.1 were exercised in other test cases.

3.4.5 FMT_SMR.2 Restrictions on security roles

3.4.5.1 Guidance Documentation

149 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:

The evaluator verified that the Common Criteria Guide contains instructions for administering the TOE both locally and remotely in section of "Administration Interfaces". By reference to the user guide, the evaluator was able to complete all tests successfully.

3.4.5.2 Tests

150 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:

There are no explicit test activities and therefore none are recorded here. All interfaces are tested throughout this test plan.

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

151 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 information was found in table 14 and section 6.5.1 of the ST.

3.5.2 FPT_APW_EXT.1 Protection of Administrator Passwords

3.5.2.1 TSS

152 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:

Passwords are described in section 6.5.2 of the ST and table 15. The password is storage as SHA1 in flash memory. No interfaces are provided to access this material directly.

3.5.3 FPT_TST_EXT.1 TSF testing

3.5.3.1 TSS

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

Findings:

The TOE self-tests are described in section 6.5.3 of the ST. The description gives sufficient evidence on how the self-tests are performed.

154 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

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

Findings:

The evaluator checked the document of Common Criteria Guide and verified that, in Section of "Power-on Self-Tests", it describes possible errors that may result from self-tests, and actions the administrator should take in response; and the description further supports the self-testing description in the TSS.

156 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

157 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.

158 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.

159 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.

160 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.

High-Level Test Description
<p>Reboot the TOE. Show that there is a record that cryptographic self-tests, integrity tests, etc. run on restart.</p> <p>Via local console, the evaluator logged in as an admin, issued on-demand self-tests, and the TOE restarted. The evaluator logged in again via local console, chose to show the results of self-tests. There was also an audit record for self-tests.</p>
Findings: PASS

3.5.4 FPT_TUD_EXT.1 Trusted Update

3.5.4.1 TSS

161 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 queried by administrators using the WEB/GUI as per section 6.5.4 of the ST. Trusted updates can be installed after as described in the process provided in section 6.5.4.

162 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:
As per section 6.5.4 of the ST, the TOE relies on a 2048-bit digital signature to ensure integrity of the software/firmware update package. Verification occurs before installation and installation fails if the signature verification fails for any reason (missing or corrupt binary or signature).

163 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:

The TOE does not select the options 'support automatic checking for updates' or 'support automatic updates'.

164 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.

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

Findings:

Certificate-based mechanisms are not used for this TOE.

166 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 TOE does not rely on hash-based integrity mechanisms.

3.5.4.2 Guidance Documentation

167 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 evaluator checked the document of Common Criteria Guide which indicates that the authenticity of the update is verified by digital signature. The evaluator also verified that section of "Software Upgrade/Downgrade and Cold Spare Upgrade Procedures" in user guide describes how to query the currently active version and how to upgrade the TOE and how to determine if upgrade is successful or unsuccessful. The description is consistent with TSS in the ST.

168 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:

See verdicts above.

169 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 TOE does not rely on hash-based integrity mechanisms.

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

Findings:

The TOE is not a distributed TOE.

171 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.

172 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:

Certificate-based mechanisms are not used for this TOE.

3.5.4.3 Tests

173 The evaluator shall perform the following tests:

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

High-Level Test Description
Get the current version of the TOE and ensure they are consistent. Install a legitimate version of the TOE. After the install, get the current version of the TOE and ensure they are consistent.
Findings: PASS

- b. [Modified by TD0477] Test 2 [conditional]: If the TOE itself verifies a digital signature to authorize the installation of an image to update the TOE the following test shall be performed (otherwise the test shall be omitted). The evaluator first confirms that no updates are pending and then performs the version verification activity to determine the current version of the product, verifying that it is different from the version claimed in the update(s) to be used in this test. The evaluator obtains or produces illegitimate updates as defined below, and attempts to install them on the TOE. The evaluator verifies that the TOE rejects all of the illegitimate updates. The evaluator performs this test using all of the following forms of illegitimate updates:
- 1) A modified version (e.g. using a hex editor) of a legitimately signed update
 - 2) An image that has not been signed
 - 3) An image signed with an invalid signature (e.g. by using a different key as expected for creating the signature or by manual modification of a legitimate signature)
 - 4) If the TOE allows a delayed activation of updates the TOE must be able to display both the currently executing version and most recently installed version. The handling of version information of the most recently installed version might differ between different TOEs depending on the point in time when an attempted update is rejected. The evaluator shall verify that the TOE handles the most recently installed version information for that case as described in the guidance documentation. After the TOE has rejected the update the evaluator shall verify, that both, current version and most recently installed version, reflect the same version information as prior to the update attempt.

High-Level Test Description
Get the current version of the TOE and ensure they are consistent.

High-Level Test Description
<p>Attempt to install a version of the TOE firmware for the following circumstances: (1) modified bit, (2) unsigned, (3) modified signature.</p> <p>Three attempts fail.</p> <p>Get the current version of the TOE using all available means and ensure they are consistent.</p> <p>The TOE does not support delayed activation.</p>
Findings: PASS

- 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). If the published hash is provided to the TOE by the Security Administrator and the verification of the hash value over the update file(s) against the published hash is performed by the TOE, then the evaluator shall perform the following tests. The evaluator first confirms that no update is pending and then performs the version verification activity to determine the current version of the product, verifying that it is different from the version claimed in the update(s) to be used in this test.
- 1) The evaluator obtains or produces an illegitimate update such that the hash of the update does not match the published hash. The evaluator provides the published hash value to the TOE and calculates the hash of the update either on the TOE itself (if that functionality is provided by the TOE), or else outside the TOE. The evaluator confirms that the hash values are different, and attempts to install the update on the TOE, verifying that this fails because of the difference in hash values (and that the failure is logged). Depending on the implementation of the TOE, the TOE might not allow the user to even attempt updating the TOE after the verification of the hash value fails. In that case the verification that the hash comparison fails is regarded as sufficient verification of the correct behaviour of the TOE
 - 2) [Modified by TD0477] The evaluator uses a legitimate update and tries to perform verification of the hash value without providing the published hash value to the TOE. The evaluator confirms that this attempt fails. Depending on the implementation of the TOE it might not be possible to attempt the verification of the hash value without providing a hash value to the TOE, e.g. if the hash value needs to be handed over to the TOE as a parameter in a command line message and the syntax check of the command prevents the execution of the command without providing a hash value. In that case the mechanism that prevents the execution of this check shall be tested accordingly, e.g. that the syntax check rejects the command without providing a hash value, and the rejection of the attempt is regarded as sufficient verification of the correct behaviour of the TOE in failing to verify the hash. The evaluator then attempts to install the update on the TOE (in spite of the unsuccessful hash verification) and confirms that this fails. Depending on the implementation of the TOE, the TOE might not allow to even attempt updating the TOE after the verification of the hash value fails. In that case the verification that the hash comparison fails is regarded as sufficient verification of the correct behaviour of the TOE
 - 3) If the TOE allows delayed activation of updates, the TOE must be able to display both the currently executing version and most recently

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

Findings:

Not Applicable - The TOE does not support published hashes.

174 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.

175 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).

Note:

The TOE only supports manual updates.

176 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

177 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 an internal battery-backed hardware clock. The internal clock may be synchronized with a time signal obtained from an external NTP server.

3.5.5.2 Guidance Documentation

178 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 evaluator checked the user guide and confirmed that it provides instructions to the administrator how to set the time by use of NTP server, and it further describes how to establish communication between the TOE and NTP server and configuration of the NTP client on the TOE to support the communication.

3.5.5.3 Tests

179 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.
- 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.

High-Level Test Description

The TOE only supports the use of an NTP server.

The evaluator configured the NTP setting on the TOE, and launched a NTP server.

The evaluator changed the time on NTP server.

There was an audit record which showed that the TOE's NTP client was in sync with updated time.

Findings: PASS

180 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:

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

3.6 TOE Access (FTA)

3.6.1 FTA_SSL_EXT.1 TSF-initiated Session Locking

3.6.1.1 Guidance Documentation

181 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:

The evaluator confirmed that the section of "Serial (CRAFT) Port Console Access and Authentication" in user guide contains instructions of configuring local admin session locking or termination.

3.6.1.2 Tests

182 The evaluator shall perform the following test:

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

High-Level Test Description
For each of 60, 90 seconds: Change the idle timeout to this value; Log into the device; With 30 seconds before the timeout expires, verify the session is still alive by sending a keep alive as described above in the TSFI commands. This should reset the timeout clock. The purpose is to ensure the timeout is not premature. Wait another minute. Verify the session is still alive by sending a keep alive. This should reset the timeout clock. The purpose is to ensure the timeout has been reset by the initial keep alive action above. Wait for the full duration of the timeout without sending any keep-alives. The session should terminate.
Findings: PASS

3.6.2 FTA_SSL.3 TSF-initiated Termination

3.6.2.1 Guidance Documentation

183 [Modified by TD0425] The evaluator shall confirm that the guidance documentation includes instructions for configuring the inactivity time period for remote administrative session termination.

Findings:
The evaluator confirmed that the section of "Viewing and Modifying System Settings" in user guide contains instructions of configuring inactivity time period for remote admin session termination.

3.6.2.2 Tests

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

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

High-Level Test Description
For each of 1, 2 minutes: Change the idle timeout to this value;

High-Level Test Description

Log into the device;
With 30 seconds before the timeout expires, verify the session is still alive by sending a keep alive as described above in the TSFI commands. This should reset the timeout clock. The purpose is to ensure the timeout is not premature.
Wait another minute. Verify the session is still alive by sending a keep alive. This should reset the timeout clock. The purpose is to ensure the timeout has been reset by the initial keep alive action above.
Wait for the full duration of the timeout without sending any keep-alives. The session should terminate.
Note that because the system uses a single command to control all idle timers, we will set in one interface and check in another.

Findings: PASS

3.6.3 FTA_SSL.4 User-initiated Termination

3.6.3.1 Guidance Documentation

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

Findings:

The evaluator verified that section of "Administration Interfaces" in Common Criteria Guide describes how to terminate local or remote interactive session.

3.6.3.2 Tests

186 For each method of remote administration, the evaluator shall perform the following tests:
a. Test 1: The evaluator initiates an interactive local session with the TOE. The evaluator then follows the guidance documentation to exit or log off the session and observes that the session has been terminated.

High-Level Test Description

Log into the serial console.
Log out using the TSFI previous discussed.
Verify that the session has been terminated.

Findings: PASS

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

High-Level Test Description

Log into the Web GUI interface.
Copy the URL presented.
Log out using the TSFI previous discussed.

High-Level Test Description
Paste the URL back into the web browser and attempt to navigate directly to it but redirected to sign it page.
Findings: PASS

3.6.4 FTA_TAB.1 Default TOE Access Banners

3.6.4.1 TSS

187 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 6.3.2 of the ST. The TOE banner is indicated on each of those mechanisms in section 6.6.4 of the ST.

3.6.4.2 Guidance Documentation

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

Findings:
The evaluator verified that the user guide describes how to configure banner message in section of “General Settings” and “Serial (CRAFT) Port Console Access and Authentication”

3.6.4.3 Tests

189 The evaluator shall also perform the following test:

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

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

3.7 Trusted path/channels (FTP)

3.7.1 FTP_ITC.1 Inter-TSF trusted channel

3.7.1.1 TSS

190 The valuator 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 for the syslog server.

3.7.1.2 Guidance Documentation

191 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 evaluator confirmed that the instructions of establishing trusted channel with remote syslog server are provided in section of "Syslog Support" of user guide; other information may be found in the section of "TLS Communication" of Common Criteria Guide.

3.7.1.3 Tests

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

193 The evaluator shall perform the following tests:

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

Note

The only trusted channel is the remote audit log which is set up as per the evaluated configuration. It is constantly tested throughout the evaluation.

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

High-Level Test Description
Engage Wireshark over the appropriate interface. Log into the Web GUI, go to the syslog setting, and reinitiate connection to remote syslog server. Examine Wireshark and verify that TLS handshake messages are captured.
Findings: PASS

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

High-Level Test Description
Engage Wireshark over the appropriate interface to capture log message traffic. Log in to the Web GUI, go to syslog setting, and send a test message. Examine Wireshark and verify that the log interface sends encrypted traffic to the remote logging server IP endpoint.
Findings: PASS

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

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

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

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

Further assurance activities are associated with the specific protocols.

High-Level Test Description
Engage Wireshark over the logging interface. Physically disconnect both the remote logging server by disconnecting the physical cable that connects the TOE into the switch. Wait 1 minute. Physically reconnect the TOE back into the switch. Examine Wireshark and verify that both the log interface continues to send encrypted Application Data packets without re-establishing a new TLS connection. Repeat the above test but wait for 15 minutes of interruption. Examine Wireshark and verify that the TOE re-establishes a new TLS connection by initiating a TLS handshake process and then send encrypted Application Data packets.

High-Level Test Description

Findings: PASS

- 194 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.
- 195 The vendor shall provide to the evaluator application layer configuration settings for all secure communication mechanisms specified by the FTP_ITC.1 requirement. This information should be sufficiently detailed to allow the evaluator to determine the application layer timeout settings for each cryptographic protocol. There is no expectation that this information must be recorded in any public- facing document or report.

Findings:

Not Applicable - The TOE is not a distributed TOE.

3.7.2 FTP_TRP.1/Admin Trusted Path

3.7.2.1 TSS

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

Findings:

Section 6.7.2 of the ST describes all of the trusted paths. The web GUI and web API are accessible over an HTTPS/TLS channel. These protocols were confirmed to be the ones specified in the requirement (Section 5.3.7 of the ST).

3.7.2.2 Guidance Documentation

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

Findings:

The evaluator confirmed that the instructions are provided in the section of "Administration Interfaces" of Common Criteria Guide.

3.7.2.3 Tests

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

Note

The only trusted path is web interface, which is set up as per the evaluated configuration. It is constantly tested throughout the evaluation.

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

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

- 199 Further assurance activities are associated with the specific protocols.
- 200 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: Not Applicable - The TOE is not a distributed TOE.
--

4 Evaluation Activities for Selection-Based Requirements

4.1 Cryptographic Support (FCS)

4.1.1 FCS_HTTPS_EXT.1 HTTPS Protocol

4.1.1.1 TSS

201 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.

4.1.1.2 Tests

202 The evaluator shall perform the following tests:

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

Note

The Web Interface traffic was already identified as TLS traffic as per FTP_TRP.1/Admin.

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

Note

Please refer to FCS_TLSS_EXT.1 for applicable test cases.

204 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.

Note

The TOE is not an HTTPS client or an HTTPS server using X.509 client authentication.

4.1.2 FCS_NTP_EXT.1 NTP Protocol

4.1.2.1 TSS

FCS_NTP_EXT.1.1

205 The evaluator shall examine the TSS to ensure identifies the version of NTP supported, how it is implemented and what approach the TOE uses to ensure the timestamp it receives from an NTP timeserver (or NTP peer) is from an authenticated source and the integrity of the time has been maintained.

Findings:

Section 6.2.9 of the ST TSS describes that the TOE uses NTP v4 and is implemented using the NTP daemon in client mode.

206 The TOE must support at least one of the methods or may use multiple methods, as specified in the SFR element 1.2. The evaluator shall ensure that each method selected in the ST is described in the TSS, including the version of NTP supported in element 1.1, the message digest algorithms used to verify the authenticity of the timestamp and/or the protocols used to ensure integrity of the timestamp.

Findings:

Section 6.2.9 of the ST TSS matches the information provided in the SFR.

4.1.2.2 Guidance Documentation

FCS_NTP_EXT.1.1

207 The evaluator shall examine the guidance documentation to ensure it provides the administrator instructions as how to configure the version of NTP supported, how to configure multiple NTP servers for the TOE's time source and how to configure the TOE to use the method(s) that are selected in the ST.

Findings:

The evaluator examined the user guide and confirmed that configuration of NTP is described in Section of "Enable authentication on any NTP servers" and "General Settings" of user guide.

FCS_NTP_EXT.1.2

208 For each of the secondary selections made in the ST, the evaluator shall examine the guidance document to ensure it instructs the administrator how to configure the TOE to use the algorithms that support the authenticity of the timestamp and/or how to configure the TOE to use the protocols that ensure the integrity of the timestamp.

Findings:

The evaluator examined the user guide and confirmed that configuration of NTP is described in Section of "Enable authentication on any NTP servers" and "General Settings" of user guide, including configuration of authentication of timestamp.

209 Each primary selection in the SFR contains selections that specify a cryptographic algorithm or cryptographic protocol. For each of these secondary selections made in the ST, the evaluator shall examine the guidance documentation to ensure that the documentation instructs the administrator how to configure the TOE to use the chosen option(s).

Findings:

The evaluator examined the user guide and confirmed that configuration of NTP is described in Section of "Enable authentication on any NTP servers" and "General Settings" of user guide, including configuration of cryptographic algorithm for timestamp authentication.

FCS_NTP_EXT.1.3

210 The evaluator shall examine the guidance documentation to ensure it provides the administrator instructions as how to configure the TOE to not accept broadcast and multicast NTP packets that would result in the timestamp being updated.

Findings:

The TOE does not accept broadcast and multicast NTP packets, so it is not configurable.

4.1.2.3 Tests

FCS_NTP_EXT.1.1

- 211 The version of NTP selected in element 1.1 and specified in the ST shall be verified by observing establishment of a connection to an external NTP server known to be using the specified version(s) of NTP. This may be combined with tests of other aspects of FCS_NTP_EXT.1 as described below.

High-Level Test Description

Engage Wireshark over the appropriate interface.

Log into the Web GUI and re-configure NTP setting

Examine Wireshark captured NTP traffic and it shows that TOE connects to NTP server and NTP server responds to TOE's request, and the NTP version is 4.

Findings: PASS

FCS_NTP_EXT.1.2

- 212 The cryptographic algorithms selected in element 1.2 and specified in the ST will have been specified in an FCS_COP SFR and tested in the accompanying Evaluation Activity for that SFR. Likewise, the cryptographic protocol selected in element 1.2 and specified in the ST will have been specified in an FCS SFR and tested in the accompanying Evaluation Activity for that SFR.
- 213 [Conditional] If the message digest algorithm is claimed in element 1.2, the evaluator will change the message digest algorithm used by the NTP server in such a way that new value does not match the configuration on the TOE and confirms that the TOE does not synchronize to this time source.
- 214 The evaluator shall use a packet sniffer to capture the network traffic between the TOE and the NTP server. The evaluator uses the captured network traffic, to verify the NTP version, to observe time change of the TOE and uses the TOE's audit log to determine that the TOE accepted the NTP server's timestamp update.
- 215 The captured traffic is also used to verify that the appropriate message digest algorithm was used to authenticate the time source and/or the appropriate protocol was used to ensure integrity of the timestamp that was transmitted in the NTP packets.

High-Level Test Description

Engage Wireshark over the appropriate interface.

On NTP server, change the time.

On the TOE, configure NTP settings to specify NTP server and message digest algorithm (SHA-1).

Observe Wireshark captured NTP traffic and check the message digest.

TOE generates an audit record for TOE's synchronization with NTP server.

On NTP server, change the NTP configuration to use different message digest algorithm (MD5).

Observe Wireshark captured NTP traffic and the TOE does not synchronize with TOE server.

TOE generates an audit record indicating that it cannot synchronize with NTP server.

Findings: PASS

FCS_NTP_EXT.1.3

216 The evaluator shall configure NTP server(s) to support periodic time updates to broadcast and multicast addresses. The evaluator shall confirm the TOE is configured to not accept broadcast and multicast NTP packets that would result in the timestamp being updated. The evaluator shall check that the time stamp is not updated after receipt of the broadcast and multicast packets.

High-Level Test Description
Engage Wireshark over the appropriate interface. On the NTP server, configure NTP server in broadcast/multicast mode only. Observe Wireshark captured traffic and it shows broadcast/multicast NTP messages from NTP server, but the TOE is be in sync with NTP server. Check TOE's corresponding audit log which reports that TOE is out of sync with NTP service.
Findings: PASS

4.1.3 FCS_TLSC_EXT.2 Extended: TLS Client Protocol with authentication

4.1.3.1 TSS

FCS_TLSC_EXT.2.1

217 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 6.2.11 of the ST TSS, the TLS client ciphersuites are defined. 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_TLSC_EXT.2.2

218 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. The evaluator shall ensure that this description identifies if certificate pinning is supported or used by the TOE and how it is implemented.

Findings:
References identifiers for the syslog audit log server are defined in section 6.2.11 of the ST TSS and can be a hostname (DNS). Both CN and SANs of the stated types are supported. Wildcards are supported in both the CN and SAN DNS type. Certificate pinning is claimed as not being supported.

FCS_TLSC_EXT.2.4

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

Findings:

Section 6.3.3 of the ST TSS claims the Supported Elliptic Curves extension is supported. The ClientHello message by default supports Elliptic Curves NIST P256, P384, P521.

FCS_TLSC_EXT.2.5

220 The evaluator shall ensure that the TSS description required per FIA_X509_EXT.2.1 includes the use of client-side certificates for TLS mutual authentication.

Findings:

In section 6.3.3 of the [ST], the author claims the "...TOE supports 2-way certificate authentication with X.509v3 certificates for Syslog communication."

4.1.3.2 Guidance Documentation

FCS_TLSC_EXT.2.1

221 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:

The evaluator checked the user guide and confirmed that the TOE does not allow administrator to configure cipher suites, which is consistent with TSS in the ST.

FCS_TLSC_EXT.2.2

222 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 evaluator checked the Common Criteria Guide (Section 3.8) and user guide (section of "Syslog Support"), and verified that it describes all identifiers, TOE's support of SAN and CN, instructions on how to configure reference identifier(s). Note that IP address is not support for syslog setting, the only case of TLS client implementation.

FCS_TLSC_EXT.2.4

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

Findings:

The evaluator checked the user guide that supported elliptic curves extension is not configurable.

FCS_TLSC_EXT.2.5

224 If the TSS indicates that mutual authentication using X.509v3 certificates is used, the evaluator shall verify that the AGD guidance includes instructions for configuring the client-side certificates for TLS mutual authentication.

Findings:

The evaluator verified that the user guide includes instructions for configuring the client-side certificates for TLS mutual authentication (see section of "Syslog Support").

4.1.3.3 Tests

FCS_TLSC_EXT.2.1

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

High-Level Test Description

Using a Lightship developed TLS server, force the TOE client to negotiate a specifically claimed ciphersuite.

Findings: PASS

226 *The goal of the following test is to verify that the TOE accepts only certificates with appropriate values in the extendedKeyUsage extension, and implicitly that the TOE correctly parses the extendedKeyUsage extension as part of X.509v3 server certificate validation.*

227 **[Modified by TD0396]** 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 extension and verify that a connection is established. The evaluator shall repeat this test using a different, but otherwise valid and trusted, certificate that lacks the Server Authentication purpose in the extendedKeyUsage extension and ensure that a connection is not established. Ideally, the two certificates should be similar in structure, the types of identifiers used, and the chain of trust.

High-Level Test Description

Construct two X.509 certificates: one with an extendedKeyUsage with 'serverAuth' and another without. Using a Lightship developed TLS server, force the TOE client to attempt a handshake with a test server and show that the X.509 certificate without the EKU fails.

Findings: PASS

228 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.

High-Level Test Description

Using a Lightship developed TLS server, force the TOE client to attempt a handshake with a test server using any of the claimed ciphersuites. The Lightship TLS server will send back an otherwise validly constructed server certificate which does not match the requested the ciphersuite.

Findings: PASS

229 Test 4: The evaluator shall configure the server to select the TLS_NULL_WITH_NULL_NULL ciphersuite and verify that the client denies the connection. Test 2 in FCS_TLSS_EXT.1.1 or FCS_TLSS_EXT.2.1 can be used as a substitute for this test.

High-Level Test Description

Using a TLS server, force the TOE client to attempt a handshake with a test server using the TLS_NULL_WITH_NULL_NULL (cipher ID 0x0000).

Findings: PASS

230

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 (for example 1.5 represented by the two bytes 03 06) and verify that the client rejects the connection.

High-Level Test Description

Using a Lightship developed TLS server, force the TOE client to attempt a handshake with a test server advertising an incorrect TLS version.

Findings: PASS

- b. 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.

High-Level Test Description

Using a Lightship developed TLS server, force the TOE client to attempt a handshake with a test server sending a modified nonce value.

Findings: PASS

- c. 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.

High-Level Test Description

Using a Lightship developed TLS server, force the TOE client to attempt a handshake with a test server sending a non-negotiated ciphersuite.

Findings: PASS

- d. If using DHE or ECDH, modify the signature block in the Server's Key Exchange handshake message, and verify that the client rejects the connection after receiving the Server Key Exchange message. 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.

High-Level Test Description

Using a Lightship developed TLS server, force the TOE client to attempt a handshake with a test server sending a mangled key exchange signature.

Findings: PASS

- e. Modify a byte in the Server Finished handshake message, and verify that the client sends an Encrypted Message followed by a FIN and ACK message. This

is sufficient to deduce that the TOE responded with a Fatal Alert and no further data would be sent.

High-Level Test Description
Using a Lightship developed TLS server, force the TOE client to attempt a handshake with a test server sending a finished message that has modified a single byte.
Findings: PASS

- f. Send a garbled message from the Server after the Server has issued the ChangeCipherSpec message and verify that the client denies the connection.

High-Level Test Description
Using a Lightship developed TLS server, force the TOE client to attempt a handshake with a test server sending a mangled finished message.
Findings: PASS

FCS_TLSC_EXT.2.2

- 231 The evaluator shall configure the reference identifier per the AGD guidance and perform the following tests during a TLS connection:
- 232 Test 1: 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.

High-Level Test Description
Force the TOE client to attempt a handshake with an OpenSSL s_server sub-application sending X.509 certificates that have the characteristics required by the test.
Findings: PASS

- 233 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.
- 234 Test 2: 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.

High-Level Test Description
Force the TOE client to attempt a handshake with an OpenSSL s_server sub-application sending X.509 certificates that have the characteristics required by the test.
Findings: PASS

- 235 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. If the TOE does mandate the presence of the SAN extension, this Test shall be omitted.

High-Level Test Description
Force the TOE client to attempt a handshake with an OpenSSL s_server sub-application sending X.509 certificates that have the characteristics required by the test.

High-Level Test Description

Findings: PASS

236 Test 4: 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.

High-Level Test Description

Force the TOE client to attempt a handshake with an OpenSSL s_server sub-application sending X.509 certificates that have the characteristics required by the test.

Findings: PASS

237 Test 5: The evaluator shall perform the following wildcard tests with each supported type of reference identifier:

1. 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.

High-Level Test Description

Force the TOE client to attempt a handshake with an OpenSSL s_server sub-application sending X.509 certificates that have the characteristics required by the test.

Findings: PASS

2. 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. 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.)

High-Level Test Description

Force the TOE client to attempt a handshake with an OpenSSL s_server sub-application sending X.509 certificates that have the characteristics required by the test.

Findings: PASS

238 Test 6: [conditional] If URI or Service name reference identifiers are supported, the evaluator shall configure the DNS name and the service identifier. The evaluator shall present a server certificate containing the correct DNS name and service identifier in the URName or SRVName fields of the SAN and verify that the connection succeeds. The evaluator shall repeat this test with the wrong service identifier (but correct DNS name) and verify that the connection fails.

Note
The TOE does not support URI or SrvName reference identifiers.

239 Test 7: [conditional] If pinned certificates are supported, the evaluator shall present a certificate that does not match the pinned certificate and verify that the connection fails.

Note

The TOE does not support pinned certificates.

FCS_TLSC_EXT.2.3

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

241 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.

Note

This test case is performed as part of FIA_X509_EXT.1.

242 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.

Note

This test case is performed as part of FIA_X509_EXT.1 and FCS_TLSC_EXT.2.2

243 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.

Note

The TOE doesn't support any override mechanism for failed certificate validation.

FCS_TLSC_EXT.2.4

244 Test 1: If using DHE or ECDH, the evaluator shall configure the server to perform an ECDHE key exchange in the TLS connection using a non-supported curve (for example P-192) and shall verify that the TOE disconnects after receiving the server's Key Exchange handshake message.

High-Level Test Description	
Force the TOE client to fail to connect to a Lightship TLS server which will use an unsupported EC curve.	
Findings: PASS	

FCS_TLSC_EXT.2.5

- 245 The purpose of these tests is to confirm that the TOE appropriately handles connection to peer servers that support and do not support mutual authentication.
- 246 Test 1: The evaluator shall establish a connection to a peer server that is not configured for mutual authentication (i.e. does not send Server's Certificate Request (type 13) message). The evaluator observes negotiation of a TLS channel and confirms that the TOE did not send Client's Certificate message (type 11) during handshake.

High-Level Test Description	
Configure a test TLS server to operate without mutual authentication and show that the TOE does not send back a certificate.	
Findings: PASS	

- 247 Test 2: The evaluator shall establish a connection to a peer server with a shared trusted root that is configured for mutual authentication (i.e. it sends Server's Certificate Request (type 13) message). The evaluator observes negotiation of a TLS channel and confirms that the TOE responds with a non-empty Client's Certificate message (type 11) and Certificate Verify (type 15) messages.

High-Level Test Description	
Configure a test TLS server to operate with mutual authentication and show that the TOE sends back a certificate.	
Findings: PASS	

4.1.4 FCS_TLSS_EXT.1 Extended: TLS Server Protocol

4.1.4.1 TSS

FCS_TLSS_EXT.1.1

- 248 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.

<p>Findings:</p> <p>In section 6.2.12 of the ST TSS, the TLS server ciphersuites are defined. 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.</p>

FCS_TLSS_EXT.1.2

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

Findings:

ST TSS section 6.2.12 explicitly states that the TOE will reject any protocol version other than TLS 1.2.

FCS_TLSS_EXT.1.3

250 [Modified by TD0450] If using ECDHE or DHE ciphers, the evaluator shall verify that the TSS describes the key agreement parameters of the server Key Exchange message.

Findings:

Section 6.2.12 of the ST TSS specifies that the TOE supports ECDHE key agreement schemes.

4.1.4.2 Guidance Documentation

FCS_TLSS_EXT.1.1

251 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:

The evaluator checked the user guide and confirmed that the TOE does not allow administrator to configure cipher suites, which is consistent with TSS in the ST.

FCS_TLSS_EXT.1.2

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

Findings:

The evaluator checked the user guide and confirmed that there is no configuration of TLS version, as it only supports TLS v1.2, which is consistent with the ST.

FCS_TLSS_EXT.1.3

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

Findings:

The evaluator checked the user guide and determined that it doesn't provide administrator's option of configuration elliptic curves.

4.1.4.3 Tests

FCS_TLSS_EXT.1.1

254 Test 1: The evaluator shall establish a TLS connection using each of the ciphersuites specified by the requirement. This connection may be established as part of the

establishment of a higher-level protocol, e.g., as part of an HTTPS session. It is sufficient to observe the successful negotiation of a ciphersuite to satisfy the intent of the test; it is not necessary to examine the characteristics of the encrypted traffic to discern the ciphersuite being used (for example, that the cryptographic algorithm is 128-bit AES and not 256-bit AES).

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

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

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

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

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

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

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

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

d. After generating a fatal alert by sending a Finished message from the client before the client sends a ChangeCipherSpec message, send a Client Hello with

the session identifier from the previous test, and verify that the server denies the connection.

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

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

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

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

High-Level Test Description
Perform a successful handshake using one of the accepted ciphersuites and verify that the Server Finished message is encrypted by validating the format of the Encrypted Handshake message.
Findings: PASS

FCS_TLSS_EXT.1.2

258

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

High-Level Test Description	
	Using a Lightship developed TLS client, connect to the TOE and attempt to negotiate SSL 2.0, SSL 3.0, TLS 1.0, TLS 1.1 and any unsupported, but otherwise valid TLS protocol versions contained in the PP.
	Findings: PASS

FCS_TLSS_EXT.1.3

- 259 If using ECDHE ciphers, the evaluator shall attempt a connection using an ECDHE ciphersuite and a configured curve. Using a packet analyser, verify that the key agreement parameters in the Key Exchange message are the ones configured. (Determining that the size matches the expected size for the configured curve is sufficient.) The evaluator shall repeat this test for each supported NIST Elliptic Curve and each supported Diffie-Hellman key size.
- 260 The evaluator shall attempt establishing connection using each claimed key establishment protocol (RSA, DH, ECDHE) with each claimed parameter (RSA key size, Diffie-Hellman parameters, supported curves) as selected in FCS_TLSS_EXT.1.3. For example, determining that the RSA key size matches the claimed size is sufficient to satisfy this test. The evaluator shall ensure that each supported parameter combination is tested.
- 261 Note that this testing can be accomplished in conjunction with other testing activities

High-Level Test Description	
	Using a Lightship developed TLS client, connect to the TOE using a valid ECDHE ciphersuite and curve combination and verify that the public key size that comes back in the Server Key Exchange message matches the expected bit size for the chosen curve.
	The TOE doesn't support DHE / RSA for key exchange.
	Findings: PASS

4.2 Identification and Authentication (FIA)

4.2.1 FIA_X509_EXT.1/Rev X.509 Certificate Validation

4.2.1.1 TSS

- 262 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).

<p>Findings:</p> <p>In the ST TSS section 6.3.6, X.509 certificates are claimed to be processed during the handshaking process for TLS. HTTPS is HTTP over TLS and is therefore included in the description implicitly.</p> <p>Regarding the check for extendedKeyUsage OIDs, the TOE will check for the OIDs it supports and expects.</p>

263

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:

In the ST TSS section 6.3.6, X.509 certificate revocation checking is performed using CRL.

264

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

Findings:

In the ST TSS section 6.3.6, X.509 certificates are checked during TLS client certificate for the purpose of user authentication. If, during the entire trust chain verification activity, any certificate under review fails a verification check, then the entire trust chain is deemed untrusted.

4.2.1.2 Tests

265

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

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

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

High-Level Test Description

Create a sequence of three X.509 certificates: a root CA, an intermediate CA signed by the root CA and a leaf node certificate signed by the intermediate CA. Load the root CA into the TOE trust store.

High-Level Test Description
Force the TOE to connect to a TLS server that sends back a certificate chain as part of the authentication process and show that the connection is accepted. Relaunch TLS server without supplying the intermediate CA certificate. Force the TOE to connect to a TLS server and show that the connection is no longer accepted.
Findings: PASS

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

High-Level Test Description
Create an expired server certificate and intermediate CA certificate. Force the TOE to connect to a TLS server with either expired server certificate or intermediate CA certificate and show both cases are not accepted. Create an expired root CA certificate. Show that loading expired root CA certificate in the trust store fails.
Findings: PASS

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

High-Level Test Description
Load the CA into the TOE trust store. Ensure the CRLs are empty. Verify that a certificate results in a successful connection. Then revoke the server certificate and place into the CRL and load into the TOE. Verify the connection now fails due to the certificate being revoked. Then modify the CRL to make the server certificate valid again and let the TOE refresh it. Revoke the intermediate CA and place into the CRL and load the CRL into the TOE. Verify the connection now fails due to the certificate being revoked. Then modify the CRL to make the intermediate certificate valid again and let the TOE refresh it. Verify that a certificate now results in a successful connection.
Findings: PASS

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

High-Level Test Description
<p>Try to load a CA certificate, which is missing the CRLSigning purpose, into the trust store. Such a attempt fails.</p> <p>Launch a TLS server with an intermediate CA certificate which is missing the CLRSigning purpose and post an empty CRL. Try to connect TOE to it, but TLS connection shall not be established.</p>
Findings: PASS

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

High-Level Test Description
<p>Force the TOE to connect to a Lightship test server which will send back a properly mangled X.509 certificate in which the ASN.1 header bytes in the first 8 bytes are modified.</p>
Findings: PASS

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

High-Level Test Description
<p>Force the TOE to connect to a Lightship test server which will send back an X.509 certificate in which the last byte of the certificate (the signature) is modified.</p>
Findings: PASS

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

High-Level Test Description
<p>Force the TOE to connect to a Lightship test server which will send back an X.509 certificate in which the public key of the certificate is modified.</p>
Findings: PASS

266 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.

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

268 For each of the following tests the evaluator shall create a chain of at least three certificates: a self-signed root CA certificate, an intermediate CA certificate and a leaf (node) certificate. The properties of the certificates in the chain are adjusted as

described in each individual test below (and this modification shall be the only invalid aspect of the relevant certificate chain).

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

High-Level Test Description
Clone the known good root CA certificate and remove the basicConstraints extension. Replace the existing known-good CA with the cloned CA. Verify the TOE fails to load the certificate.
Clone a known good intermediate CA certificate and remove the basicConstraints extension. Launch a TLS server with such a modified intermediate CA certificate. Verify that TOE fails to establish TLS connection to TLS server.
Findings: PASS

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

High-Level Test Description
Clone the known good root CA certificate and set its basicConstraints extension to FALSE. Replace the existing known-good CA with the cloned CA. Verify the TOE fails to load the certificate.
Clone a known good intermediate CA certificate and set its basicConstraints extension to FALSE. Launch a TLS server with such a modified intermediate CA certificate. Verify that TOE fails to establish TLS connection to TLS server.
Findings: PASS

269 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).

Note:
This functionality was tested for TLSC, but not for TLSS because only FCS_TLSS_EXT.1 is claimed.

4.2.2 FIA_X509_EXT.2 X.509 Certificate Authentication

4.2.2.1 TSS

270 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 central certificate store for certificates to be stored and examined as part of the validation process.

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

Findings:

Section 6.3.7 in the ST TSS indicates that if a CRL cannot be obtained, the validation will fail.

4.2.2.2 Tests

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

273 The evaluator shall demonstrate that using a valid certificate that requires certificate validation checking to be performed in at least some part by communicating with a non-TOE IT entity. The evaluator shall then manipulate the environment so that the TOE is unable to verify the validity of the certificate, and observe that the action selected in FIA_X509_EXT.2.2 is performed. If the selected action is administrator-configurable, then the evaluator shall follow the guidance documentation to determine that all supported administrator-configurable options behave in their documented manner.

High-Level Test Description

The tester stopped posting CRL. Any attempt to validate a certificate failed.

Findings: PASS

4.2.3 FIA_X509_EXT.3 Extended: X509 Certificate Requests**4.2.3.1 TSS**

274 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 developer did not select "device specific information".

4.2.3.2 Guidance Documentation

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

Findings:

The evaluator checked the user guide and confirmed that description of X509 certificate requests is described in section of “General Settings” (for web server certificate) and section of “Syslog Support” (for syslog client certificate)

4.2.3.3 Tests

275 The evaluator shall perform the following tests:

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

High-Level Test Description

Using the TOE CSR generator, create a new CSR and download to an external CA entity for signing. Using OpenSSL, verify that the information in the CSR is as expected.

Findings: PASS

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

High-Level Test Description

The CSR from the previous test is signed by a CA. Try to import newly generated certificate into the TOE without supply full CA certificate chain. The certificate cannot be imported because the CA is missing. Then concatenate the newly created leaf certificate with all CA certificates in the CA certificate chain and attempt to reimport. The import is successful.

Findings: PASS

4.3 Security management (FMT)

4.3.1 FMT_MOF.1/Functions Management of security functions behaviour

4.3.1.1 TSS

276 For distributed TOEs see chapter 4.4.1.1. There are no specific requirements for non-distributed TOEs.

Findings:

The TOE is not a distributed TOE.

4.3.1.2 Tests

277 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.

High-Level Test Description	
	Sign in to the TOE as a non-admin user via Web UI, attempt to change syslog settings, but fail – Modification of syslog settings is disabled.
	Findings: PASS

- 278 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.
- 279 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.

High-Level Test Description	
	Sign in to the TOE as an admin via Web GUI, attempt to change syslog settings and show that the change is permitted. Verify the effect of the change.
	Findings: PASS

- 280 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.

Note:
The TOE does not claim this functionality and this test is not applicable.

- 281 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.

282 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.

Note:

The TOE does not claim this functionality and this test is not applicable.

283 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.

Note:

The TOE does not claim this functionality and this test is not applicable.

284 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.

285 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.

Note:

The TOE does not claim this functionality and this test is not applicable.

286 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.

Note:

The TOE does not claim this functionality and this test is not applicable.

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

tests. The attempt(s) to determine the behaviour of the selected functions with administrator authentication shall be successful.

Note:

The TOE does not claim this functionality and this test is not applicable.

4.3.2 FMT_MTD.1/CryptoKeys Management of TSF Data

4.3.2.1 TSS

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

Findings:

The TOE is not a distributed TOE.

4.3.2.2 Tests

289 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.

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

High-Level Test Description

As a non-admin user, attempt to generate a private key using the CSR generation functionality and show it cannot succeed.

As an admin user, attempt to generate a private key using the CSR generation functionality and show it does succeed.

Findings: PASS

5 Vulnerability Assessment

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.

The developer shall provide documentation identifying the list of software and hardware components³ that compose the TOE. Hardware components should identify at a minimum the processors used by the TOE. Software components include applications, the operating system and other major components that are independently identifiable and reusable (outside the TOE) such as a web server and protocol or cryptographic libraries. This additional documentation is merely a list of the name and version number of the components, and will be used by the evaluators in formulating hypotheses during their analysis.

Findings:

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

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 following sources of public vulnerabilities were considered in formulating the specific list of flaws to be investigated by the evaluators, as well as to reference in directing the evaluators to perform key-word searches during the evaluation of the TOE. Hypothesis sources for public vulnerabilities were:

- Keysight security advisories (<https://about.keysight.com/en/quality/security/advisory.shtml> and <https://support.ixiacom.com/support-services/security-advisories>)
- NIST National Vulnerabilities Database <https://web.nvd.nist.gov/view/vuln/search>
- NIST National Vulnerabilities Database (can be used to access CVE and US-CERT databases identified below): <https://web.nvd.nist.gov/view/vuln/search>
- Common Vulnerabilities and Exposures: <http://cve.mitre.org/cve/>
<https://www.cvedetails.com/vulnerability-search.php>
- US-CERT: <http://www.kb.cert.org/vuls/html/search>
- Exploit / Vulnerability Search Engine: www.exploitsearch.net
- SecurITeam Exploit Search: www.securiteam.com

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

- Tenable Network Security <http://nessus.org/plugins/index.php?view=search>
- Tipping Point Zero Day Initiative <http://www.zerodayinitiative.com/advisories>
- Offensive Security Exploit Database: <https://www.exploit-db.com/>
- Rapid7 Vulnerability Database: <https://www.rapid7.com/db/vulnerabilities>
- OpenSSL Vulnerabilities: <https://www.openssl.org/news/vulnerabilities.html>
- Google

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

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

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

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