Fortinet

FortiProxy 1.0

Assurance Activity Report

Version 1.1

August 2019

Document prepared by

www.lightshipsec.com

Table of Contents

1 Introduction

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

1.1 Evaluation Identifiers

Table 1: Evaluation Identifiers

Scheme	Canadian Common Criteria Scheme
Evaluation Facility	Lightship Security
Developer/Sponsor	Fortinet
TOE	FortiProxy 1.0
Security Target	Fortinet FortiProxy 1.0 Security Target v1.0
Protection Profile	collaborative Protection Profile for Network Devices, v2.0e+20180314 (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 v3.1R4
Evaluation Methodology	CEM v3.1R4
Supporting Documents	Evaluation Activities for Network Device cPP, v2.0e+20180314 (NDcPP-SD)

2 TOE Details

2.1 Overview

1 Fortinet FortiProxy is a secure web proxy that protects employees against internet-borne attacks by incorporating multiple detection techniques such as web filtering, DNS filtering, data loss prevention, antivirus, intrusion prevention and advanced threat protection. It helps enterprises enforce internet compliance using granular application control. High-performance physical and virtual appliances deploy on-site to serve small, medium and large enterprises.

2.2 TOE Models

Model	CPU	RAM	ASIC
FPX-400E	Intel i7-4790S	8GB	CP9
FPX-2000E	Intel E5-2680v4	64GB	CP9
FPX-4000E	Dual Intel E5-2680v4	128GB	CP9

2

2.3 Reference Documents

Table 3: List of Reference Documents

Ref	Document
[ST]	Fortinet FortiProxy 1.0 Security Target v1.0
[AGD-1]	FortiProxy 1.0 FIPS 140-2 and Common Criteria Technote
[AGD-2]	Fortinet FortiProxy Administration Guide Version 1.0.0
[AGD-3]	FIPS 140-2 Non-Proprietary Security Policy FortiProxy-400E/2000E/4000E v0.3
[AGD-4]	Fortinet FortiOS – Log Reference v5.6.7

2.4 Summary of SFRs

Table 4: List of SFRs

Requirement	Title
FAU_GEN.1	Audit Data Generation
FAU_GEN.2	User Identity Association

Requirement	Title
FAU_STG_EXT.1	Protected Audit Event Storage
FCS_CKM.1	Cryptographic Key Generation
FCS_CKM.2	Cryptographic Key Establishment
FCS_CKM.4	Cryptographic Key Destruction
FCS_COP.1/DataEncryption	Cryptographic Operation (AES Data Encryption/Decryption)
FCS_COP.1/SigGen	Cryptographic Operation (Signature Generation and Verification)
FCS_COP.1/Hash	Cryptographic Operation (Hash Algorithm)
FCS_COP.1/KeyedHash	Cryptographic Operation (Keyed Hash Algorithm)
FCS_HTTPS_EXT.1	HTTPS Protocol
FCS_RBG_EXT.1	Random Bit Generation
FCS_SSHS_EXT.1	SSH Server Protocol
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
FMT_MTD.1/CryptoKeys	Management of TSF Data

Requirement	Title
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

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:	The required information was found in the TSS table 13 in the [ST] in section 6 for FAU_GEN.1.

4 For distributed TOEs the evaluator shall examine the TSS to ensure it describes which auditable events are generated and recorded by which TOE components. 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 [NIAP TD0410] 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:	[AGD-1] “Log Specific Settings” lists PP audit requirements. [AGD-4] provides a comprehensive log reference

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:	[AGD-1] “Log Specific Settings” specifies that all configuration changes, including failures, are logged by default when configured in FIPS-CC mode.

3.1.1.3 Tests

7 The evaluator shall test the TOE’s ability to correctly generate audit records by having the TOE generate audit records for the events listed in the table of audit events and administrative actions listed above. This should include all instances of an event: for

instance, if there are several different I&A mechanisms for a system, the FIA_UIA_EXT.1 events must be generated for each mechanism. The evaluator shall test that audit records are generated for the establishment and termination of a channel for each of the cryptographic protocols contained in the ST. If HTTPS is implemented, the test demonstrating the establishment and termination of a TLS session can be combined with the test for an HTTPS session. When verifying the test results, the evaluator shall ensure the audit records generated during testing match the format specified in the guidance documentation, and that the fields in each audit record have the proper entries.

Findings:	These tests are conducted throughout the test plan.

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

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

Findings:	The TOE is not a distributed TOE.

3.1.2 FAU_GEN.2 User identity association

3.1.2.1 Tests

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

11 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

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

Findings:	The required information was found in the [ST] TSS Section 6, Table 13 for FAU_STG_EXT.1.

13 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:	The required information was found in the [ST] TSS Section 6, Table 13 for FAU_STG_EXT.1.

14 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:	The required information was found in the [ST] TSS Section 6, Table 13 for FAU_STG_EXT.1.

15 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:	The required information was found in the [ST] TSS Section 6, Table 13 for FAU_STG_EXT.1.

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

Findings:	The TOE is not a distributed TOE.

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

Findings:	The TOE is not a distributed TOE.

3.1.3.2 Guidance Documentation

18 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:	[AGD-1] references [AGD-2] Logging, which provides information on establishing a secure channel to a FortiAnalyzer device for external logging.

19 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:	[AGD-1] "Log Specific Settings". *"Log messages are cached on the local Fortinet unit before being offloaded to the remote FortiAnalyzer device. The log messages are cached on the local disk or in system memory if the unit does not have disk storage. The log message cache is separate and distinct from local log storage."* [ST] TSS states that logs are cached then sent in near real-time.

20 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:	[ST] TSS and [AGD-1] in combination with [AGD-2] correspond with respect to logging configurations.

3.1.3.3 Tests

21 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 v3.5.6. The evaluator always pulled remote auditing records from test cases unless explicitly stated otherwise. In this way, the successful execution of a test case in this test plan confirms that correct reception of the necessary audit records outlined in FAU_GEN.1 above.

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
Log into the device and show that log entries are stored on the local system via the "Event Log" panel. Show that only the configured number of log files are kept on the device at any given time. Show that when log files are dropped, the oldest events are dropped.
Findings: PASS

High-Level Test Description
Log into the device and show that log entries are stored on the local system via the "Event Log" panel and that they are sourced from the memory log instead of the disk log. Ensure that once the log is filled, the oldest logs are dropped off.
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 Algorithm Validations

22 Due to the new algorithm numbering schemes used by the CAVP, we provide a mapping table to show how the validated modules map to the claimed services, which we can map to the specific algorithms needed to meet a requirement.

Module Name	CAVP Cert	Services in the scope of the evaluation
Fortinet FortiProxy FIPS Cryptographic Library v1.0	C703, C832	Password hashing

Module Name	CAVP Cert	Services in the scope of the evaluation
Fortinet FortiProxy RBG Cryptographic Library v1.0	C658, C799	DRBG
Fortinet FortiProxy SSL Cryptographic Library v1.0	C655, C702, C787, C806	TLS SSH Trusted update signature verification
Fortinet CP9 Cryptographic Library v5.6	C813	Hardware acceleration of TLS and SSH related cryptographic operations.

Findings:	The evaluators considered all of the certificates and found evidence that each of the claimed processor architectures are present for each of the claimed algorithms and cryptographic services. This work was supported by a spreadsheet to help keep track of the certificates, architectures and claimed module. The hardware acceleration of the CP9 ASIC works if the algorithm is implemented by the ASIC and the functionality is enabled by administrative configuration, then the ASIC will perform the cryptographic function. If the algorithm is not implemented by the ASIC or if the functionality is disabled by administrative configuration, then the firmware implementation will perform the cryptographic function.

3.2.2 FCS_CKM.1 Cryptographic Key Generation

3.2.2.1 TSS

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

Findings:	The requested information is found in table 13 in section 6 of the [ST] for FCS_CKM.1. Key sizes and scheme usages are described.

3.2.2.2 Guidance Documentation

24 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:	Key generation schemes automatically configured when put into FIPS-CC mode.

3.2.2.3 Tests

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

Key Generation for FIPS PUB 186-4 RSA Schemes

26 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*.

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

a) Random Primes:

- Provable primes
- Probable primes

b) Primes with Conditions:

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

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

Findings:	CAVP certificate numbers are described in section 3.2.1. For RSA key generation, this is validated by CAVP C702 and C806.

Key Generation for Elliptic Curve Cryptography (ECC)

FIPS 186-4 ECC Key Generation Test

29 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

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

Findings:	CAVP certificate numbers are described in section 3.2.1. For ECC key generation, this is validated by CAVP C813.

Key Generation for Finite-Field Cryptography (FFC)

31 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.

32 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

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

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

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

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

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

36 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.

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

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

38 for each FFC parameter set and key pair.

39 [NIAP TD0291] *Testing for FFC Schemes using Diffie-Hellman group 14 is done as part of testing in CKM.2.1.*

Findings:	No FFC key generation is performed by the TOE.

3.2.3 FCS_CKM.2 Cryptographic Key Establishment

3.2.3.1 TSS

40 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 (including whether the TOE acts as a sender, a recipient, or both). If Diffie-Hellman group 14 is selected from FCS_CKM.2.1, the TSS shall describe how the implementation meets RFC 3526 Section 3.

Findings:	The requested information is found in table 13 in section 6 of the [ST] for FCS_CKM.2. Each scheme's usage is identified, however also included RSA which wasn't

	described in the SFR in section 5 [ST]. See OR1.4. OR1.4 was resolved by removing RSA from FCS_CKM.2 in the TSS.

3.2.3.2 Guidance Documentation

41 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:	Key establishment scheme(s) automatically configured when put into FIPS-CC mode.

3.2.3.3 Tests

Key Establishment Schemes

42 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

43 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

44 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.

45 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.

46 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.

47 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.

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

Validity Test

49 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.

50 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).

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

Findings:	CAVP certificate numbers are described in section 3.2.1. For elliptic-key based key exchange, this is validated as per CAVP 702 and 806 (KAS-ECC Component).

RSA-based key establishment schemes

Technical Decisions:	The following assurance activities have been modified by TD0402.

52 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.

Findings:	The evaluator conducted testing using an independent known-good implementation during test cases for FCS_TLSS_EXT.1.1 and FCS_TLSC_EXT.2.1 using RSA public/private keys. The connections were successful.

Diffie-Hellman Group 14

53 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:	The evaluator conducted testing using an independent known-good implementation during test cases for FCS_TLSS_EXT.1.1, FCS_SSHS_EXT.1.7 and FCS_IPSEC_EXT.1 when using DH group 14. The connections were successful.

3.2.4 FCS_CKM.4 Cryptographic Key Destruction

3.2.4.1 TSS

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

Findings:	Section 6.1 of the [ST] describes each key type, its origin, storage mode(s) and how it is securely erased when it is no longer required for various use cases. The evaluator considered the various trusted paths and channels and means to authenticate security administrators included in the scope of the evaluation and found appropriate descriptions for keys and CSPs for each of those mechanisms.

55 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:	Section 6.1 of the [ST] describes how each plaintext CSP in non-volatile storage is overwritten with zeros on factory reset. During factory reset the entire flash memory device is zeroed as per table 13 in section 6 for FCS_CKM.4.

56 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 does not use "destruction of reference" to erase CSPs.

57 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:	Section 6.1 indicates that all CSPs and keys are stored in plaintext format on the Flash memory.

58 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

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

the Guidance Documentation for description of the detail of such cases where destruction may be prevented or delayed.

Findings:	The TSS does not describe any circumstances or evaluated configurations in which the key destruction does not occur as described. It is, however, imperative, that a properly educated administrator zero out the device prior to altering its operational configuration or when decommissioning it. Fortinet provides careful instructions in their Guidance Documentation regarding such practices.

59 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:	As described in section 6.1 of the [ST], the TOE overwrites all CSPs with zeros.

3.2.4.2 Guidance Documentation

60 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.

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

Findings:	[AGD-1] provides guidance on zeroizing keys if required, and [ST] A.TRUSTED_ADMINISTRATOR ensure admins are appropriately trained for key zeroization.

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

3.2.5.1 Tests

AES-CBC Known Answer Tests

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

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

63 **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.

64 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.

65 **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.

66 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.

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

68 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.

69 **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].

70 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

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

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

AES-CBC Monte Carlo Tests

73 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]
```

74 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.

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

Findings:	CAVP certificate numbers are described in section 3.2.1. AES-CBC cryptographic operations are validated under CAVP C655, CAVP C703, CAVP C787 or CAVP C832 respectively.

AES-GCM Test

76 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.

77 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.

78 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.

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

Findings:	CAVP certificate numbers are described in section 3.2.1. AES-GCM cryptographic operations are validated under CAVP C806, CAVP C702, CAVP C703 or CAVP C832 respectively.

AES-CTR Known Answer Tests

Technical Decisions:	The following assurance activities have been modified by TD0397.

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

81 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.

82 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.

83 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.

84 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].

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

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

AES-CTR Multi-Block Message Test

86 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

87 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:

88 # Input: PT, Key

89 for i = 1 to 1000:

90 CT[i] = AES-ECB-Encrypt(Key, PT) PT = CT[i]

91 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:	The TOE does not use AES-CTR mode.

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

3.2.6.1 Tests

ECDSA Algorithm Tests

ECDSA FIPS 186-4 Signature Generation Test

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

Findings:	CAVP certificate numbers are described in section 3.2.1. ECDSA SigGen operations are validated under CAVP C813.

ECDSA FIPS 186-4 Signature Verification Test

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

Findings:	CAVP certificate numbers are described in section 3.2.1. ECDSA SigVer operations are validated under CAVP C813, C703, C832, C702, and C806.

RSA Signature Algorithm Tests

Signature Generation Test

94 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.

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

Findings:	CAVP certificate numbers are described in section 3.2.1. RSA SigGen operations are validated under CAVP C813, C702 and C806.

Signature Verification Test

96 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.

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

Findings:	CAVP certificate numbers are described in section 3.2.1. RSA SigVer operations are validated under CAVP C813, C702 and C806.

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

3.2.7.1 TSS

98 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:	Table 13 in section 6 of the [ST] for FCS_COP.1/Hash indicates how SHS is used for each TOE need for cryptographic hashing services.

3.2.7.2 Guidance Documentation

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

Findings:	Though not explicitly stated, it is believed putting the TOE into FIPS-CC mode automatically configures required hash sizes.

3.2.7.3 Tests

100 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.

101 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

102 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

103 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

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

Selected Long Messages Test - Byte-oriented Mode

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

Pseudorandomly Generated Messages Test

106 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:	CAVP certificate numbers are described in section 3.2.1. Secure hashing services are validated under CAVP C813, C702, C806, C703, and C832.

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

3.2.8.1 TSS

107 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 requested information was found in a table as part of table 13 in section 6 of the [ST] for FCS_COP.1/KeyedHash.

3.2.8.2 Tests

108 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:	CAVP certificate numbers are described in section 3.2.1. Keyed hashing services are validated under CAVP C813, C702, C806, C703, and C832.

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

109 Documentation shall be produced—and the evaluator shall perform the activities—in accordance with Appendix D of NDcPPe.

3.2.9.1 TSS

110 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:	Table 13 in section 6 of the [ST] provides this information for the row labelled FCS_RBG_EXT.1. The 256-bit seed contains full entropy post conditioning.

3.2.9.2 Guidance Documentation

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

Findings:	[AGD-1]. Ensure TOE is in FIPS-CC mode.

3.2.9.3 Tests

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

113 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).

114 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.

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

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

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

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

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

Findings:	CAVP certificate numbers are described in section 3.2.1. CAVP certificate number C658 and C799 covers all claimed platforms for the CTR-DRBG.

116

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:	Table 13 in section 6 of the [ST] for FIA_AFL.1 provides the necessary details. The TOE provides for time-based account unlocking only.

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:	Table 13 in section 6 of the [ST] for FIA_AFL.1 indicates that the local serial console is immune from being locked out.

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:	[ST] identifies both GUI and CLI options to configure failed authentication parameters, with guidance on how to do so detailed in [AGD-2]. Timeout is 5mins by default after 3 failed authentication attempts.

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:	Local console admin access can never be locked out.

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.

High-Level Test Description
Using the local console, set the administrator threshold to 3 attempts. Change the duration to 1 minute. Logout of the local console. Using the SSH interface, log into the TOE twice using an incorrect password. On the third attempt, log in correctly and verify that the threshold has not been reached. Using the SSH interface, log into the TOE three times using an incorrect password. On the fourth attempt, log in correctly and verify that the threshold has been reached and that the user cannot log in. Using a secondary workstation with a distinct IP, log into the TOE using SSH with the correct password. The attempt should fail. Attempt to log into the local console using the admin account. The attempt should succeed. Wait 1 minute. Repeat the above test using the Web GUI interface instead of the SSH interface.
Findings: PASS

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

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

Note:	The TOE only claims time-based lockout.

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

High-Level Test Description
Set the threshold to 3 minutes. Using the SSH interface, log into the TOE three times using an incorrect password. On the fourth attempt, log in correctly and verify that the threshold has been reached and that the user cannot log in. Start a timer. Wait 2m40s seconds. Attempt to login correctly over the Web GUI interface. The attempt should fail. Wait another 40 seconds (3m20s total) to give the system time to unlock the mechanism and settle down. Attempt to login correctly over the SSH interface. The attempt should succeed. Attempt to login correctly over the Web GUI interface. The attempt should succeed. Repeat the above test case for durations 5 minutes. Failed reauthentication attempts occur 20 seconds before the timer is expected to expire and 20 seconds after the timer is expected to expire.
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:

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

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

Findings:	[AGD-1] "The FIPS-CC Mode of Operation" specifies password requirements, and [AGD-2] "Adding a new Administrator" shows the process for entering that in.

3.3.2.2 Tests

123 The evaluator shall perform the following tests.

a) Test 1: The evaluator shall compose passwords that either meet the requirements, or fail to meet the requirements, in some way. For each password, the evaluator shall verify that the TOE supports the password.

While the evaluator is not required (nor is it feasible) to test all possible compositions of passwords, the evaluator shall ensure that all characters, and a minimum length listed in the requirement are supported, and justify the subset of those characters chosen for testing.

High-Level Test Description
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:	Each login process is described in detail in table 13 in section 6 of the [ST] for FIA_UIA_EXT.1/FIA_UAU_EXT.2 for CLI (serial and SSHv2) and Web GUI 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:	The set of actions that are permitted prior to I&A are found in table 13 in section 6 of the [ST] for FIA_UIA_EXT.1/FIA_UAU_EXT.2. Both local and remote administration mechanisms are discussed.

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:	[AGD-1] "The FIPS-CC Mode of Operation".

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 device does not have any services configured prior to I&A other than a TOE banner.
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 and being able to view the version of the TOE.
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 TOE is not a distributed TOE.

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.

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

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.

Findings:	This test case is covered in FPT_TUD_EXT.1.

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:	The information pertaining to activities available prior to I&A can be found in table 13 in section 6 of the [ST] for FIA_UIA_EXT.1/FIA_UAU_EXT.2. The same table for row FMT_MTD.1/CoreData describes the roles used by the TOE to prevent access to various TSF data by non-administrative users.

3.4.3.2 Guidance Documentation

141 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:	[AGD-2] “Administrative Profiles” describes different administrator roles, and per NDcPP, and Adminisator is considered a Security Admin..

3.4.4 FMT_SMF.1 Specification of Management Functions

142 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)

143 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:	In table 13 in section 6 of the [ST] for the row labelled FMT_SMF.1, it indicates that all functions can be performed over all interfaces.

144 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

145 See section 4.4.4.1.

3.4.4.3 Tests

146 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

147 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:	[AGD-1] “Remote Access Requirements”. Secure remote admin enabled in FIPS-CC mode. Both [AGD-1] and [AGD-2] offer guidance to administer the TOE locally and remotely.

3.4.5.2 Tests

148 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

149 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:	All pre-shared, symmetric and private keys are described in the table in section 6.1 of the [ST]. This table further describes how each of the keys are stored. For those that are not stored in plaintext, the mechanism by which it is rendered opaque is given. Section 6.1 indicates that there are no interfaces specifically designed to view key data in the clear.

3.5.2 FPT_APW_EXT.1 Protection of Administrator Passwords

3.5.2.1 TSS

150 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:	All authentication data are described in the table in section 6.1 of the [ST]. This table further describes how each of the data elements are stored. For those that are not stored in plaintext, the mechanism by which it is rendered opaque is given. Section 6.1 indicates that there are no interfaces specifically designed to view authentication credentials in the clear.

3.5.3 FPT_TST_EXT.1 TSF testing

3.5.3.1 TSS

151 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:	A reasonable treatment of the TOE self-tests is described In table 13 in section 6 of the [ST] for the row labelled FPT_TST_EXT.1.

152 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

153 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:	[AGD-1] “Self Tests” and “Error Mode” describe self-tests and error mode, however do not mention possible errors that may occur. [AGD-3] describes error messages and associated actions for self test

154 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

155 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.

156 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.

157 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.

158 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
Force a reboot of the TOE using the Web interface. Show that there is a record that cryptographic self-tests and integrity tests run on restart.
Findings: PASS

3.5.4 FPT_TUD_EXT.1 Trusted Update

3.5.4.1 TSS

159 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:	In table 13 in section 6 of the [ST] in the table row labelled FPT_TUD_EXT.1, the ST author describes how the current active firmware can be queried (‘status’ in the CLI or ‘System Information’ via the web GUI). The TOE does not support delayed activation.

160 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:	In table 13 in section 6 of the [ST] in the table row labelled FPT_TUD_EXT.1, the ST provides the necessary information requested. TOE updates are digitally signed and the process by which this digital signature is verified is described. The actions for successful and unsuccessful verification is described.

161 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:	As described in table 13 in section 6 of the [ST] for the table row labelled FPT_TUD_EXT.1, the TOE automatically checks for updates but does not automatically install them. The actions involved in the automatic check are described.

162 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.

163 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:	The TOE does not claim certificates for the verification of the digital signature.

164 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 claim published hash as a firmware integrity mechanism.

3.5.4.2 Guidance Documentation

165 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:	[AGD-1] "Installing the FIPS-CC firmware build". Per [ST], the TOE will automatically check for updates by default, and notify administrators of availability.

166 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:	[AGD-1] “Tusted Update” includes guidance for obtaining and checking MD5 checksums for verification of initial firmware build, however these are only a secondary (not cryptographically secure) measure. 2048 bit RSA signatures are used for verification of updates.

167 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:	[AGD-1] includes guidance for obtaining and checking MD5 checksums for updates, however only as a secondary measure. [AGD-1] shows the use of 2048 bit RSA signatures for verification.

168 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.

169 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.

170 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:	N/A.

3.5.4.3 Tests

171 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 using all available means and ensure they are consistent. Install a legitimate version of the TOE for the following circumstances: a downgrade, a same-grade. After the install, get the current version of the TOE using all available means and ensure they are consistent.
Findings: PASS

b) Test 2 (if digital signatures are used): 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 using all available means and ensure they are consistent. Attempt to install a version of the TOE firmware for the following circumstances: (a) a downgrade, (b) a same-grade in which the image has been modified accordingly: (1) modified bit, (2) unsigned, (3) modified signature. After the install, get the current version of the TOE using all available means and ensure they are consistent.
Findings: PASS

c) Test 3 (if published hash is verified on the TOE): 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

Test Not Applicable	The TOE does not support published hashes.

2) The evaluator uses a legitimate update and tries to perform verification of the hash value without storing the published hash value on the TOE. The evaluator confirms that this attempt fails. Depending on the implementation of the TOE it might not be possible to attempt the verification of the hash value without providing a hash value to the TOE, e.g. if the hash value needs to be handed over to the TOE as a parameter in a command line message and the syntax check of the command prevents the execution of the command without providing a hash value. In that case the mechanism that prevents the execution of this check shall be tested accordingly, e.g. that the syntax check rejects the command without providing a hash value, and the rejection of the attempt is regarded as sufficient verification of the correct behaviour of the TOE in failing to verify the hash. The evaluator then attempts to install the update on the TOE (in spite of the unsuccessful hash verification) and confirms that this fails. Depending on the implementation of the TOE, the TOE might not allow to even attempt updating the TOE after the verification of the hash value fails. In that case the verification that the hash comparison fails is regarded as sufficient verification of the correct behaviour of the TOE

Test Not Applicable	The TOE does not support published hashes.

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.

Test Not Applicable	The TOE does not support published hashes or delayed activation.

172 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.

173 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. The test cases above are not applicable to automatic checking of updates, since there are no images to install during an automatic check.

174 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

175 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:	In the TSS in table 13 in section 6 of the [ST] for the row labelled FPT_STM_EXT.1, the ST author indicates the various subcomponents of the TOE which rely on the time. This section further describes how the time is maintained during operation.

3.5.5.2 Guidance Documentation

176 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:	[AGD-2] “Settings” provides instructions to set the time. NTP not included in scope.

3.5.5.3 Tests

177 The evaluator shall perform the following tests:

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

High-Level Test Description
Get the current date and time. Change the date/time in the past by 1 day, 1 hour and 42 minutes. Verify the date/time was set properly.

High-Level Test Description
Change the date/time in the future by 7 days, 1 hour and 42 minutes. Verify the date/time was set properly.
Findings: PASS

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

Test Not Applicable	The TOE does not claim NTP.

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

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

3.6 TOE Access (FTA)

3.6.1 FTA_SSL_EXT.1 TSF-initiated Session Locking

3.6.1.1 Guidance Documentation

179 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:	[AGD-2] “Settings”. Inactivity session termination supported, with timeout options between 1-480 mins. Per [ST], can be configured by Administrator with no restrictions on time.

3.6.1.2 Tests

180 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 1, 3, 5 minutes: Change the idle timeout to this value;

High-Level Test Description
Log into the device; Wait for the full duration of the timeout. 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

3.6.2 FTA_SSL.3 TSF-initiated Termination

3.6.2.1 Guidance Documentation

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

Findings:	[AGD-2] "Settings". Inactivity session termination supported, with timeout options between 1-480 mins. Per [ST], can be configured by Administrator with no restrictions on time.

3.6.2.2 Tests

182 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, 3, 5 minutes: Change the idle timeout to this value; Log into the device; Wait for the full duration of the timeout. 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

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

Findings:	[AGD-1] Terminating Local and Remote Administration Sessions.

3.6.3.2 Tests

184 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 SSH CLI interface.
Log out using the TSFI previous discussed.
Verify that the session has been terminated.
Log into the Web interface.
Copy the URL presented.
Log out using the TSFI previous discussed.
Paste the URL back into the web browser and attempt to navigate to it and show it is not permitted.
Findings: PASS

3.6.4 FTA_TAB.1 Default TOE Access Banners

3.6.4.1 TSS

185 [NIAP TD0338] 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 in table 13 in section 6 of the [ST] for the row labelled FTA_TAB.1 indicates that the TOE banner is available at both the CLI and the Web GUI. Though trivial, there is no mention of any differences between CLI over serial connection and SSH. OR1.5. Additionally, the requirement for the TSS stating that the TOE is displaying an advisory notice and a consent warning message for each administrative method

	of access has not been met. OR1.5 was raised and resolved with updates to [ST] section 6.

3.6.4.2 Guidance Documentation

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

Findings:	[AGD-1] “Admin access disclaimer” describes pre-configured access banner, with instructions on how to enable.

3.6.4.3 Tests

187 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 SSH CLI interface. Change the banner to a random string. Log into fresh sessions for all interactive interfaces and show that the banner was modified and is presented prior to I&A. 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

188 [NIAP TD0290] The evaluator shall examine the TSS to determine that, for all communications with authorized IT entities identified in the requirement, each secure communication mechanism is identified in terms of the allowed protocols for that IT entity, whether the TOE acts as a server or a client, and the method of assured identification of the non-TSF endpoint. The evaluator shall also confirm that all secure communication mechanisms are described in sufficient detail to allow the evaluator to match them to the cryptographic protocol Security Functional Requirements listed in the ST.

Findings:	The TSS in table 13 in section 6 of the [ST] for the row labelled FTP_ITC.1 describes all of the trusted channels. All trusted channels use TLS as a secure transport. Details about how the endpoints are assured are provided in the same table for the row labelled FCS_TLSC_EXT.2 (found in section 5.3.2 of the [ST]) and FIA_X509_EXT.1 (found in 5.3.3 of the [ST]). OR1.6 was raised concerning the

	requirement of the designation of TOE as client or server for Inter-TSF trusted channel. OR 1.6 resolved in [ST] section 6.

3.7.1.2 Guidance Documentation

189 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:	[AGD-1] via enabling FIPS-CC mode. How a connection should be re-established. Is described in [AGD-1] “Reconnecting to FortiAnalyzer”.

3.7.1.3 Tests

190 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 channels 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 CLI and disable and re-enable the logging interface. Examine wireshark and verify that the log interface sends a CLIENT HELLO TLS message.
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 into the serial device and logout. Examine wireshark and verify that the log interface sends encrypted traffic to the remote logging server IP endpoint.
Findings: PASS

d) [NIAP TD0290] *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.*

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

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

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

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

High-Level Test Description
Engage wireshark over the logging interface. Perform a looping login activity once per second to ensure logging messages are being tested continuously. Physically disconnect the remote logging server (disconnect from the remote end rather that from the TOE end to ensure that the TOE is unable to invoke any layer 2 carrier-sensing mechanism). Wait 5 seconds. Physically reconnect the remote logging server. Examine wireshark and verify that the log interface continues to send encrypted Application Data packets. Repeat the above with a 30 minute timeout performing a series of every 30 seconds instead.
Findings: PASS

191 Further assurance activities are associated with the specific protocols.

192 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.

Test Not Applicable	The TOE is not a distributed TOE.

3.7.2 FTP_TRP.1/Admin Trusted Path

3.7.2.1 TSS

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

Findings:	The TSS in table 13 in section 6 of the [ST] for the row labelled FTP_TRP.1 describes all of the trusted paths. The CLI is remotely accessible over an SSH channel and the web GUI is accessible over an HTTPS/TLS channel. These protocols were confirmed to be included in the set of requirements: FCS_HTTPS_EXT.1, FCS_TLSS_EXT.1 and FCS_SSHS_EXT.1 (all found in section 5.3.2 of the [ST]).

3.7.2.2 Guidance Documentation

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

Findings:	[AGD-1] “Remote access requirements”. FIPS-CC mode.

3.7.2.3 Tests

195 The evaluator shall perform the following tests:

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

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

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

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

c) [NIAP TD0290] Test 3: *This test is removed.*

196 Further assurance activities are associated with the specific protocols.

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

Findings:	Refer to section **Error! Reference source not found. Error! Reference source not found.**.

4 Evaluation Activities for Optional Requirements

4.1 Security management (FMT)

4.1.1 FMT_MTD.1/CryptoKeys Management of TSF Data

4.1.1.1 TSS

198 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.

4.1.1.2 Tests

199 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.

200 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
Create an unprivileged user. As the unprivileged user, attempt to generate a private key using the CSR generation functionality and show it cannot succeed. As the privileged user, attempt to generate a private key using the CSR generation functionality and show it does succeed.
Findings: PASS

5 Evaluation Activities for Selection-Based Requirements

5.1 Cryptographic Support (FCS)

5.1.1 FCS_HTTPS_EXT.1 HTTPS Protocol

5.1.1.1 TSS

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 TSS in table 13 in section 6 of the [ST] for the row labelled FCS_HTTPS_EXT.1 describes at a reasonably high level how the TOE adheres to RFC 2818.

5.1.1.2 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, and the evaluator shall perform the following test:

a) Test 1: The evaluator shall demonstrate that using a certificate without a valid certification path results in an application notification. Using the administrative guidance, the evaluator shall then load a valid certificate and certification path, and demonstrate that the function succeeds. The evaluator then shall delete one of the certificates, and show that the selection listed in the ST occurs.

Test Not Applicable	The TOE does not make use of certificates in its capacity for FCS_TLSS_EXT.1 and therefore this test is not applicable.

5.1.2 FCS_SSHS_EXT.1 SSH Server

5.1.2.1 TSS

FCS_SSHS_EXT.1.2

205 [NIAP_TD0339] The evaluator shall check to ensure that the TSS contains a description of the public key algorithms that are acceptable for use for authentication and that this list conforms to FCS_SSHS_EXT.1.5. and ensure that if password-based authentication methods have been selected in the ST then these are also described.

Findings:	The TSS in table 13 in section 6 of the [ST] for the row labelled FCS_SSHS_EXT.1 describes that the acceptable public key algorithms is restricted to ssh-rsa.

FCS_SSHS_EXT.1.3

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

Findings:	The TSS in table 13 in section 6 of the [ST] for the row labelled FCS_SSHS_EXT.1 describes both the size of the large packet how the TOE reacts when it receives one (it drops it).

FCS_SSHS_EXT.1.4

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

Findings:	The TSS in table 13 in section 6 of the [ST] for the row labelled FCS_SSHS_EXT.1 does not describe any optional characteristics. The TOE only supports AES-CBC (with varying key sizes) which match those given in FCS_SSHS_EXT.1.4 in section 5.3.2.

FCS_SSHS_EXT.1.5

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

Findings:	The TSS in table 13 in section 6 of the [ST] for the row labelled FCS_SSHS_EXT.1 does not describe any optional characteristics. The TOE only supports ssh-rsa which match those given in FCS_SSHS_EXT.1.5 in section 5.3.2.

FCS_SSHS_EXT.1.6

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

Findings:	The TSS in table 13 in section 6 of the [ST] for the row labelled FCS_SSHS_EXT.1 lists the supported integrity algorithms which match those given in FCS_SSHS_EXT.1.6 in section 5.3.2.

FCS_SSHS_EXT.1.7

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

Findings:	The TSS in table 13 in section 6 of the [ST] for the row labelled FCS_SSHS_EXT.1 lists the supported key exchange algorithm which matches that given in FCS_SSHS_EXT.1.7 in section 5.3.2.

FCS_SSHS_EXT.1.8

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

1. Both thresholds are checked by the TOE.

2. Rekeying is performed upon reaching the threshold that is hit first.

212 [NIAP TD0281] The intention of FCS_SSHC_EXT.1.8 and FCS_SSHS_EXT.1.8 SFRs is to ensure that the TOE implements both thresholds. The NIT also acknowledges that it is possible that hardware limitation may prevent reaching data transfer threshold in less than one hour. In cases where data transfer threshold could not be reached due to hardware limitations it is acceptable to omit testing of this (SSH rekeying based on data transfer threshold) threshold if both the following conditions are met:

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

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

Findings:	The TSS in table 13 in section 6 of the [ST] for the row labelled FCS_SSHS_EXT.1 claims the required volume and time-based thresholds for rekeying. Both thresholds are checked and careful emphasis is made on the fact that the volume-based check is against the aggregate of the sending and receiving data (as required in [PP] Application Note 102. No mention is made of threshold not being reached due to hardware limitations.

5.1.2.2 Guidance Documentation

FCS_SSHS_EXT.1.4

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

Findings:	This is automatically configured when put into FIPS-CC mode. [AGD-1] FIPS-CC mode.

FCS_SSHS_EXT.1.5

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

Findings:	This is automatically configured when put into FIPS-CC mode. [AGD-1] FIPS-CC mode.

FCS_SSHS_EXT.1.6

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

Findings:	This is automatically configured when put into FIPS-CC mode. [AGD-1] FIPS-CC mode.

FCS_SSHS_EXT.1.7

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

Findings:	This is automatically configured when put into FIPS-CC mode. [AGD-1] FIPS-CC mode.

FCS_SSHS_EXT.1.8

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

Findings:	This is automatically configured when put into FIPS-CC mode. [AGD-1] FIPS-CC mode.

5.1.2.3 Tests

FCS_SSHS_EXT.1.2

218 [NIAP TD0339] Test 1: If password-based authentication methods have been selected in the ST then using the guidance documentation, the evaluator shall configure the TOE to accept password-based authentication, and demonstrate that user authentication succeeds when the correct password is provided by the user.

Note	This test was conducted as part of FIA_UIA_EXT.1/FIA_UAU_EXT.2.

219 [NIAP TD0339] Test 2: If password-based authentication methods have been selected in the ST then the evaluator shall use an SSH client, enter an incorrect password to attempt to authenticate to the TOE, and demonstrate that the authentication fails.

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

Note	This test was conducted as part of FIA_UIA_EXT.1/FIA_UAU_EXT.2.

FCS_SSHS_EXT.1.3

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

High-Level Test Description
Using a custom SSH client, log into the TOE using a valid username and password, but ensure that a large packet is transmitted and verify the connection is terminated.
Findings: PASS

FCS_SSHS_EXT.1.4

221 The evaluator must ensure that only claimed ciphers and cryptographic primitives are used to establish a SSH connection. To verify this, the evaluator shall start session establishment for a SSH connection from a remote client (referred to as 'remote

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

High-Level Test Description
Using SSH client, log into the TOE using each of the claimed ciphers in turn and show that the communication is successful. Review the negotiation line from the server to ensure that there are no additional ciphers claimed by the implementation that differ from the ST or the PP requirements.
Findings: PASS

FCS_SSHS_EXT.1.5

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

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

High-Level Test Description
Using SSH client, log into the TOE using each of the claimed public key algorithms with a valid key and show that the communication is successful.
Findings: PASS

224 [NIAP TD0412] *Test objective: The purpose of this negative test is to verify that the server rejects authentication attempts of clients that present a public key that does not match public key(s) associated by the TOE with the identity of the client (i.e. the public keys are unknown to the server). To demonstrate correct functionality it is sufficient to determine that an SSH connection was not established after using a valid username and an unknown key of supported type.*

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

High-Level Test Description
Create two public/private key pairs. Load the public key portion from pair A into the TOE. Using SSH client, log into the TOE using private key from pair B. The attempt should fail.

High-Level Test Description
Findings: PASS

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

High-Level Test Description
Create a public/private key pair for DSA unsupported algorithms. Load the public key portion from the newly generated key into the TOE for the admin user. The attempt to load may fail. Using SSH client, log into the TOE using newly generated private key portion. The attempt should fail.
Findings: PASS

FCS_SSHS_EXT.1.6

227 [NIAP TD0337] Test 1: (conditional, if an HMAC or AEAD_AES_*_GCM algorithm is selected in the ST) The evaluator shall establish an SSH connection using each of the algorithms, except “implicit”, specified by the requirement. It is sufficient to observe (on the wire) the successful negotiation of the algorithm to satisfy the intent of the test.

228 [NIAP TD0337] Note: To ensure the observed algorithm is used, the evaluator shall ensure a non-aes*- gcm@openssh.com encryption algorithm is negotiated while performing this test.

High-Level Test Description
Using SSH client, log into the TOE using each of the claimed integrity algorithms in turn and show that the communication is successful. Review the negotiation line from the server to ensure that there are no additional integrity algorithms claimed by the implementation that differ from the ST or the PP requirements.
Findings: PASS

229 [NIAP TD0337] Test 2: (conditional, if an HMAC or AEAD_AES_*_GCM algorithm is selected in the ST) The evaluator shall configure an SSH client to only allow a MAC algorithm that is not included in the ST selection. The evaluator shall attempt to connect from the SSH client to the TOE and observe that the attempt fails.

230 [NIAP TD0337] Note: To ensure the proposed MAC algorithm is used, the evaluator shall ensure a non-aes*- gcm@openssh.com encryption algorithm is negotiated while performing this test.

High-Level Test Description
Using SSH client, log into the TOE using each the hmac-md5 integrity algorithm and show that the communication is unsuccessful.
Findings: PASS

FCS_SSHS_EXT.1.7

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

High-Level Test Description
Using SSH client, log into the TOE using diffie-hellman-group-1-sha1 key exchange algorithm and show that the communication is unsuccessful.
Findings: PASS

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

High-Level Test Description
Using SSH client, log into the TOE using each of the claimed key exchange algorithm and show that the communication is successful.
Findings: PASS

FCS_SSHS_EXT.1.8

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

234 [NIAP TD0336] For testing of the time-based threshold the evaluator shall use an SSH client to connect to the TOE and keep the session open until the threshold is reached. The evaluator shall verify that the SSH session has been active longer than the threshold value and shall verify that the TOE initiated a rekey (the method of verification shall be reported by the evaluator).

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

High-Level Test Description
Using a custom SSH client, log into the TOE and push at least 1GB of data in less than 1 hour to force rekeying. Show that the TOE rekeys before the 1GB of data is reached or the 1 hour time limit is reached.
Findings: PASS

236 [NIAP TD0336] The evaluator shall verify that more data has been transmitted within the SSH session than the threshold allows and shall verify that the TOE initiated a rekey (the method of verification shall be reported by the evaluator).

237 The evaluator shall verify that more data has been transmitted within the SSH session than the threshold allows and a corresponding audit event has been generated by the TOE.

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

High-Level Test Description
Using a custom SSH client, log into the TOE and push less than the rekey limit of data in at least 1 hour to force rekeying by time-based mechanisms.
Findings: PASS

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

Note	These limits are not configurable for this TOE.

5.1.3 FCS_TLSC_EXT.2 Extended: TLS Client Protocol with authentication

5.1.3.1 TSS

FCS_TLSC_EXT.2.1

240 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:	The TSS in table 13 in section 6 of the [ST] for the row labelled FCS_TLSC_EXT.2 identifies the versions of TLS supported as well as the ciphersuites supported. The ciphersuites identified in the TSS are identical to those claimed in the SFR in section 5.3.2. It should be noted that each of the identified trusted channels have identical configurations even though they are distinct implementations. That is, the evaluator considered whether there should have been iterations of FCS_TLSC_EXT.2 per channel and found that the [ST] was accurately portraying the collective as a single SFR. This finding applies to all TSS assurance activities for FCS_TLSC_EXT.2.

FCS_TLSC_EXT.2.2

241 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:	The TSS in table 13 in section 6 of the [ST] for the row labelled FCS_TLSC_EXT.2 describes how the reference identifiers for the FAZ logging server is established. The TOE claims both IP addresses and hostnames. Due to the fact that Test 5.2 in section 4.1.9.3 is not optional, wildcards are, in fact, supported by the TOE for hostnames. The TSS indicates certificate pinning is not supported.

242 Note that where a TLS channel is being used between components of a distributed TOE for FPT_ITT.1, the requirements to have the reference identifier established by the user are relaxed and the identifier may also be established through a "Gatekeeper" discovery process. The TSS should describe the discovery process and highlight how the reference identifier is supplied to the "joining" component.

Findings:	The TOE is not a distributed TOE.

FCS_TLSC_EXT.2.4

243 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:	The TSS in table 13 in section 6 of the [ST] for the row labelled FCS_TLSC_EXT.2 provides the requested information. The supported curves are not configurable.

FCS_TLSC_EXT.2.5

244 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 table 13 in section 6, row FCS_TLSC_EXT.2 of the [ST], the author claims the "…TOE supports presentation of an X.509v3 client certificate for authentication as required by the FAZ Audit Server."

5.1.3.2 Guidance Documentation

FCS_TLSC_EXT.2.1

245 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:	No further configuration is needed to ensure the TLS client conforms with the description in the TSS.

FCS_TLSC_EXT.2.2

246 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 CLI is needed to set the reference identifier of the FortiAnalyzer as described in [AGD-1] section "FortiAnalyzer configuration". The reference identifier is placed in the "server" parameter. The [CLI] guide describes the "server" parameter more fully in the "log fortianalyzer" section. The reference identifier can be an IP address or FQDN.

FCS_TLSC_EXT.2.4

247 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:	No further configuration is needed to ensure the TLS client conforms with the description in the TSS.

FCS_TLSC_EXT.2.5

248 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 [AGD-1] client certificate is set using the “certificate” option in the “log fortianalyzer” configuration tree. The process for generating or loading this certificate can be found in the [AGD-2] guide Chapter “Certificates”.

5.1.3.3 Tests

FCS_TLSC_EXT.2.1

249 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

250 [NIAP TD0396] *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*.

251 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

252 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

253 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

254 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.

High-Level Test Description
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) [NIAP TD0289] 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

255 Note that where a TLS channel is being used between components of a distributed TOE for FPT_ITT.1, the requirements to have the reference identifier established by the user are relaxed and the identifier may also be established through a "Gatekeeper" discovery process.

Test Not Applicable	The TOE is not a distributed TOE.

256 The evaluator shall configure the reference identifier per the AGD guidance and perform the following tests during a TLS connection:

a) [NIAP TD0257] 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.

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.

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

b) 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

c) [NIAP TD0257] 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.
Findings: PASS

d) 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

e) 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.come) 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

f) 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 URIName 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 URL or SrvName reference identifiers.

g) 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

257 Test 1: The evaluator shall demonstrate that using a certificate without a valid certification path results in the function failing. Using the administrative guidance, the evaluator shall then load a certificate or certificates needed to validate the certificate to be used in the function and demonstrate that the function succeeds. If the certificate is validated and a trusted channel is established, the test passes. The evaluator then shall delete one of the certificates and show that the certificate is not validated and the trusted channel is not established.

High-Level Test Description
Force the TOE client to attempt a handshake with an OpenSSL s_server sub-application sending a leaf certificate without the Intermediate CA to complete the chain. Show this fails. Then, resend the leaf and Intermediate CA certificates and show that the channel is established successfully.
Findings: PASS

FCS_TLSC_EXT.2.4

258 Test 1: If using ECDHE ciphers, 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

Technical Decisions: The following assurance activities have been modified by TD0256.

259 *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.*

260 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

261 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

5.1.4 FCS_TLSS_EXT.1 Extended: TLS Server Protocol

5.1.4.1 TSS

FCS_TLSS_EXT.1.1

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

Findings:	The TSS in table 13 in section 6 of the [ST] for the row labelled FCS_TLSS_EXT.1 identifies the versions of TLS supported as well as the ciphersuites supported. The ciphersuites identified in the TSS are identical to those claimed in the SFR in section 5.3.2.

FCS_TLSS_EXT.1.2

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

Findings:	The TSS in table 13 in section 6 of the [ST] for the row labelled FCS_TLSS_EXT.1 claims all unsupported TLS and SSL versions are rejected.

FCS_TLSS_EXT.1.3

264 The evaluator shall verify that the TSS describes the key agreement parameters of the server Key Exchange message.

Findings:	The TSS in table 13 in section 6 of the [ST] for the row labelled FCS_TLSS_EXT.1 states the following: "*The TLS server is capable of negotiating ciphersuites that include DHE, and ECDHE key agreement schemes. The DHE key agreement parameters are restricted to 2048 bits and are hardcoded into the server.*" In this case, because the TOE only supports DHE and ECDHE key agreement schemes (it does not support RSA key exchange), the wording is accurate. For DHE key agreement, the TOE must provide both parameters *p* and *g* in the TLS Server Key Exchange message. The most important parameter is the size of *p* which frames the overall security strength of the scheme. By stating that the DHE parameters are restricted to 2048 bits, the ST is claiming *p* is going to be 2048 bits in size. For ECDHE, the parameters are intrinsically defined by the curves in use, which the ST claims in the TSS is restricted to NIST P-256.

5.1.4.2 Guidance Documentation

FCS_TLSS_EXT.1.1

265 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:	[AGD-1] FIPS-CC mode.

FCS_TLSS_EXT.1.2

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

Findings:	[AGD-1] FIPS-CC mode.

FCS_TLSS_EXT.1.3

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

Findings:	[AGD-1] FIPS-CC mode.

5.1.4.3 Tests

FCS_TLSS_EXT.1.1

268 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

269 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

270 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

271 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) [NIAP TD0342] Test Intent: The intent of this test is to ensure that the server's TLS implementation immediately makes use of the key exchange and authentication algorithms to:

a) Correctly encrypt (D)TLS Finished message

b) Encrypt every (D)TLS message after session keys are negotiated

Test 4 e): 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

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

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

FCS_TLSS_EXT.1.3

273 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.

274 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.

275 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. Using a Lightship developed TLS client, connect to the TOE using a valid DHE ciphersuite and verify that the DH parameters that come back in the Server Key Exchange message matches the expected bit size. Using a Lightship developed TLS client, connect to the TOE using a valid RSA ciphersuite and verify that the public key modulus that comes back in the Server Certificate message matches the expected bit size.
Findings: PASS

5.2 Identification and Authentication (FIA)

5.2.1 FIA_X509_EXT.1/Rev X.509 Certificate Validation

5.2.1.1 TSS

276 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 sufficient to verify the status of a X.509 certificate only when it's loaded onto the device. It is not necessary to verify the revocation status of X.509 certificates during power-up self-tests (if the option for using X.509 certificates for self-testing is selected).

Findings:	The TSS in table 13 in section 6 of the [ST] for the row labelled FIA_X509_EXT.1 provides the points at which the certificates are checked for validity. The evaluator considered the various ways in which the TOE's claimed functions might affect this verdict: - For FCS_TLSS_EXT.1, the TOE is not responsible for validating certificates due to the nature of the SFR; - For FCS_TLSC_EXT.2, the TOE is responsible for validating certificates from the external entities in the TOE's environment; - For administrative use, the TOE will need to validate any CAs imported into the TOE's trust store necessary to provide appropriate root-of-trust for the certificate chain validations; - For administrative use, the TOE will need to validate a certificate response resulting from a certificate signing request (CSR) as required by FIA_X509_EXT.3; - For administrative use, the TOE will need to validate any consumer-obtained certificates that might be used for the TOE's web GUI server being served over HTTPS;

- The TOE does not claim X.509 for SSH authentication; and

- The TOE does not support nor claim certificate-based checks for trusted updates or self-tests.

Each of these scenarios is presented by the [ST] in the TSS section for FIA_X509_EXT.1.

In addition, the row labelled FIA_X509_EXT.1 in the TSS in table 13 in section 6 describes the various extendedKeyUsage rules which are not checked because their functionality is not claimed.

Finally, the TSS indicates that certificate revocation is checked each time the certificate is used by the TOE for authentication. The TOE uses CRLs for this purpose.

5.2.1.2 Tests

277 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:

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 certificate to be used in the function, and shall use this chain to demonstrate that the function succeeds.

Test 1b: The evaluator shall then delete one of the certificates in the presented chain (i.e. the root CA certificate or other intermediate certificate, but not the end-entity certificate), and show that an attempt to validate an incomplete 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. 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. Remove the root CA from the TOE trust store. 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 X.509 certificate with a ‘notAfter’ date in the past. Force the TOE to connect to a TLS server that sends back this certificate and show it is not accepted. Show that CA certificates in the trust store that expire after being loaded result in an error.

High-Level Test Description
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.

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
Load a CA into the trust store that is missing the CRLSigning purpose. Load the CRL for the corresponding CA. Show that the TOE prevents loading this CRL.
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
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.

High-Level Test Description
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
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.
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
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.
Findings: PASS

278 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.

279 [NIAP TD0228] The goal of the following tests it to verify that the TOE accepts only certificates that have been marked as CA certificates by using basicConstraints with the CA flag set to True (and implicitly that the TOE correctly parses the basicConstraints extension as part of X509v3 certificate chain validation).

280 [NIAP TD0228] 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) [NIAP TD0228] 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 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.
Findings: PASS

b) [NIAP TD0228] 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 CA certificate and set the basicConstraints extension to have the CA flag set to FALSE. Replace the existing known-good CA with the cloned CA. Verify the load fails.
Findings: PASS

c) [NIAP TD0228] *Test 3: This test is no longer required.*

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

Findings:	This functionality was tested for TLSC and TLSS.

5.2.2 FIA_X509_EXT.2 X.509 Certificate Authentication

5.2.2.1 TSS

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

Findings:	The TSS in table 13 in section 6 of the [ST] for the row labelled FIA_X509_EXT.2 indicates that the TOE stores trusted certificates in a trust store. Appropriate instructions for loading certificates into the trust store for validation can be found in the guidance documentation.

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

Findings:	The TSS in table 13 in section 6 of the [ST] for the row labelled FIA_X509_EXT.2 indicates that the certificate is not accepted if the certificate cannot be validated

	(including if revocation checks fail). As previously described in the AA for FCS_TLSC_EXT.2, each of the claimed trusted channels have the same behaviour: there are no distinctions to be made between them for X.509 validation purposes. The default action is not claimed to be user-configurable and no evidence was found in the guidance documentation to the contrary.

5.2.2.2 Tests

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

285 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 disabled the CRL distribution point and reset the TOE to remove any cached CRLs. The tester noted that the TOE did not allow the connection when the CRL was not available. This is consistent with the selection in FIA_X509_EXT.2.2.
Findings: PASS

5.2.3 FIA_X509_EXT.3 Extended: X509 Certificate Requests

5.2.3.1 TSS

286 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 TSS in table 13 in section 6 of the [ST] for the row labelled FIA_X509_EXT.3 provides the required fields for generating a certificate signing request (CSR). Device-specific information is not among the mandatory or optional information requested of the user when generating a CSR.

5.2.3.2 Guidance Documentation

287 [NIAP TD0333] The evaluator shall check to ensure that the guidance documentation contains instructions on requesting certificates from a CA, including generation of a Certification Requests. 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:	[AGD-2] “Certificates”.

5.2.3.3 Tests

288 The evaluator shall perform the following tests:

a) [NIAP TD0333] Test 1: The evaluator shall use the guidance documentation to cause the TOE to generate a Certification Request. The evaluator shall capture the generated request and ensure that it conforms to the format

specified. The evaluator shall confirm that the Certification Request provides the public key and other required information, including any necessary user-input information.

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) [NIAP TD0333] Test 2: The evaluator shall demonstrate that validating a response message to a Certification Request without a valid certification path results in the function failing. The evaluator shall then load a certificate or certificates as trusted CAs needed to validate the response message, and demonstrate that the function succeeds.

High-Level Test Description
The CSR from the previous test is signed by a CA which is not yet loaded in the TOE trust store. It is imported into the TOE. The certificate cannot be imported because the CA is missing. Then add the CA to the trust store and attempt to reimport. The import is successful.
Findings: PASS

5.3 Security management (FMT)

5.3.1 FMT_MOF.1/Functions Management of security functions behaviour

5.3.1.1 TSS

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

Note:	The TOE is not a distributed TOE.

5.3.1.2 Tests

290 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
Create an unprivileged user. For each of the defined TSFI functions found in the TOE, attempt to change them one at a time to one of their legal values and show that the change is not permitted.
Findings: PASS

291 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.

High-Level Test Description
For each of the defined TSFI functions found in the TOE, attempt to change them one at a time using the privileged ‘admin’ user to one of their legal values and show that the change is permitted. Verify the effect of the change.
Findings: PASS

292 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.

293 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 will not be conducted.

294 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.

Note:	The TOE does not claim this functionality and this test will not be conducted.

295 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.

296 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 will not be conducted.

297 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.

298 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 will not be conducted.

299 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 will not be conducted.

300 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 will not be conducted.

6 Vulnerability Assessment

301 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.

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

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

303 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: - Fortinet security advisories (https://fortiguard.com/psirt) - 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 - Community (Symantec) security community: https://www.securityfocus.com/ - US-CERT: http://www.kb.cert.org/vuls/html/search - Tenable Network Security http://nessus.org/plugins/index.php?view=search

[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.

- 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

- Google

Type 1 Hypothesis searches were conducted on June 20, 2019 and included the following search terms:

- Fortinet;

- FortiProxy;

- Linux kernel;

- TLS;

- OpenSSH;

- Apache;

- TCP

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.