Fortinet, Inc.

FortiAnalyzer 6.2

Assurance Activity Report

Version 1.6

December 2022

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, Inc.
TOE	FortiAnalyzer 6.2.8 Build: 9598
Security Target	Fortinet FortiAnalyzer 6.2 Security Target, v1.7
Protection Profile	collaborative Protection Profile for Network Devices, Version 2.2e, 23-March-2020

1.2 Evaluation Methods

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

Table 2: Evaluation Methods

Evaluation Criteria	CC v3.1R5
Evaluation Methodology	CEM v3.1R5
Supporting Documents	Supporting Document Mandatory Technical Document Evaluation Activities for Network Device cPP, December-2019, Version 2.2
Interpretations	**NDcPP v2.2e** TD0527: Updates to Certificate Revocation Testing (FIA_X509_EXT.1) TD0528: NIT Technical Decision for Missing EAs for FCS_NTP_EXT.1.4 *NTP not claimed* TD0536: NIT Technical Decision for Update Verification Inconsistency TD0537: NIT Technical Decision for Incorrect reference to FCS_TLSC_EXT.2.3

TD0538: NIT Technical Decision for Outdated link to allowed-with list
TD0546: NIT Technical Decision for DTLS - clarification of Application Note 63 *The TOE does not claim FCS_DTLS_EXT.1.*
TD0547: NIT Technical Decision for Clarification on developer disclosure of AVA_VAN
TD0555: NIT Technical Decision for RFC Reference incorrect in TLSS Test
TD0556: NIT Technical Decision for RFC 5077 question
TD0563: NiT Technical Decision for Clarification of audit date information
TD0564: NiT Technical Decision for Vulnerability Analysis Search Criteria
TD0569: NIT Technical Decision for Session ID Usage Conflict in FCS_DTLSS_EXT.1.7
TD0570: NiT Technical Decision for Clarification about FIA_AFL.1
TD0571: NiT Technical Decision for Guidance on how to handle FIA_AFL.1
TD0572: NiT Technical Decision for Restricting FTP_ITC.1 to only IP address identifiers
TD0580: NIT Technical Decision for clarification about use of DH14 in NDcPPv2.2e
TD0581: NIT Technical Decision for Elliptic curve-based key establishment and NIST SP 800-56Arev3
TD0591: NIT Technical Decision for Virtual TOEs and hypervisors
TD0592: NIT Technical Decision for Local Storage of Audit Records
TD0631: NIT Technical Decision for Clarification of public key authentication for SSH Server
TD0632: NIT Technical Decision for Consistency with Time Data for vNDs
TD0633: NIT Technical Decision for IPsec IKE/SA Lifetimes Tolerance *The TOE does not claim FCS_IPSEC_EXT.1.*

	TD0634: NIT Technical Decision for Clarification required for testing IPv6 TD0635: NIT Technical Decision for TLS Server and Key Agreement Parameters TD0636: NIT Technical Decision for Clarification of Public Key User Authentication for SSH *The TOE does not claim FCS_SSHC_EXT.1.* TD0638: NIT Technical Decision for Key Pair Generation for Authentication TD0639: NIT Technical Decision for Clarification for NTP MAC Keys TD0670: NIT Technical Decision for Mutual and Non-Mutual Auth TLSC Testing
Tools	Please see associated Test Plan.

1.3 Reference Documents

Table 3: List of Reference Documents

Ref	Document
PP	collaborative Protection Profile for Network Devices, Version 2.2e, 23-March-2020
SD	Supporting Document Mandatory Technical Document Evaluation Activities for Network Device cPP, December-2019, Version 2.2
ST	Fortinet FortiAnalyzer 6.2 Security Target, v1.7
SUPP	NDcPP Common Criteria and FIPS 140-2 Technote, FortiAnalyzer 6.2, version 1.7, December 12, 2022
QSG	Fortinet FortiAnalyzer Hardware Guides: • FortiAnalyzer 800F QuickStart Guide, Doc No. 05-560-443238-20200902 • FortiAnalyzer 1000F QuickStart Guide, Doc No 02-60-559470-20210125 • FortiAnalyzer 2000E Information, Doc No. 05-540-294825-20210105 • FortiAnalyzer 3000F Information, Doc No. 05-540-293346-20170905 • FortiAnalyzer 3500G QuickStart Guide, January 08, 2020 • FortiAnalyzer 3700F Information, Doc No. 05-541-382365-20210105
CLI	Fortinet FortiAnalyzer 6.2.8 CLI Reference, PDF Doc No. 05-628-539013-20210513
EVT	Fortinet FortiManager & FortiAnalyzer - Event Log Reference, Version 6.2.7, No. 02-627-549533-20201118

Ref	Document
ADMIN	Fortinet FortiAnalyzer 6.2.8 Administration Guide, PDF Doc No. 05-628-547780-20210901

2 Evaluation Activities for SFRs

2.1 Security Audit (FAU)

2.1.1 FAU_GEN.1 Audit data generation

2.1.1.1 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:	[ST] Section 6.1.1 – The following information is logged as a result of generating/importing, changing or deleting cryptographic keys: a) Generate CSR. Action and key reference. b) Import Certificate. Action and key reference. c) Import CA Certificate. Action and key reference.

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

Findings:	The TOE is not a distributed TOE.

2.1.1.2 Guidance Documentation

5 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:	[EVT] Contains information on the audit logs that can be emitted from the TOE.

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

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

	vendor. The evaluator reviewed the contents of the documentation and looked specifically for functionality related to the scope of the evaluation. Where there was missing or incomplete descriptions for the functionality such that the user could not complete the testing AAs, the evaluator requested the vendor to supply augmented guidance information. In the end, the vendor provided a more comprehensive guidance "supplement" document in the form of [SUPP].

2.1.1.3 Tests

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

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

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

High-Level Test Description
The audit records are generated in tests conducted throughout the test plan. The evaluator compared the audit records against those in the [EVT] and found them to be consistent. The TOE is not a distributed TOE.
Findings: PASS

2.1.2 FAU_GEN.2 User identity association

2.1.2.1 TSS & Guidance Documentation

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

2.1.2.2 Tests

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

Findings:	Please refer to test cases which support FAU_GEN.1.1.

12 For distributed TOEs the evaluator shall verify that where auditable events are instigated by another component, the component that records the event associates the event with the identity of the instigator. The evaluator shall perform at least one

test on one component where another component instigates an auditable event. The evaluator shall verify that the event is recorded by the component as expected and the event is associated with the instigating component. It is assumed that an event instigated by another component can at least be generated for building up a secure channel between two TOE components. If for some reason (could be e.g. TSS or Guidance Documentation) the evaluator would come to the conclusion that the overall TOE does not generate any events instigated by other components, then this requirement shall be omitted.

Findings:	The TOE is not a distributed TOE.

2.1.3 FAU_STG_EXT.1 Protected audit event storage

2.1.3.1 TSS

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

Findings:	[ST] Section 6.1.3 – The TOE is configured to transmit log data to another FortiAnalyzer platform. Trusted channel is provided via TLS.

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

Findings:	[ST] Section 6.1.3 – The amount of disk space used for logs is configurable but dependent on the total available TOE disk space. Section 6.1.3 of the [ST] also states that when the audit store is full, the TOE begins to overwrite the oldest logs. Lastly it says only authorized administrators may view audit records and there is no capability to modify the logs.

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

Findings:	The TOE is standalone and not a distributed TOE.

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

Findings:	[ST] Section 6.1.3 – When the local audit store is full, the TOE will begin overwriting the oldest logs.

17 The evaluator shall examine the TSS to ensure that it details whether the transmission of audit information to an external IT entity can be done in real-time or periodically. In case the TOE does not perform transmission in real-time the evaluator needs to verify that the TSS provides details about what event stimulates the transmission to be made as well as the possible as well as acceptable frequency for the transfer of audit data.

Findings:	[ST] Section 6.1.3 – Logs can be sent in real time or according to a schedule defined by the administrator.

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

Findings:	The TOE is not a distributed TOE.

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

Findings:	The TOE is not a distributed TOE.

2.1.3.2 Guidance Documentation

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

Findings:	[SUPP] “FortiAnalyzer configuration” section describes the settings that need to be applied to set up the remote audit server.

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

Findings:	[SUPP] Page 19 “Log Specific Settings” section - Log messages are cached on the local FortiAnalyzer 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.

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

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

2.1.3.3 Tests

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

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

Note: Verification that the data is encrypted is satisfied by FTP_ITC.1 for the logging channel. The logging server is FAZ-1000E running firmware version v6.2.0-build1374 200916 as described in the Test Setup. The logging server is not the TOE. Due to the log-forwarding mechanism used on logging server, the audit records are therefore confirmed to have been successfully received by the audit server whenever the test cases are run.

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

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

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

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

High-Level Test Description
Through CLI, set a threshold for disk size for log storage. When that disk size is reached, the TOE shall delete oldest logs to create space for new log files. Fill the storage until the threshold is reached (with dummy files and audit logs). Verify that after the threshold is reached, the oldest logs are removed, while new logs are still being generated. Compare the status of the log files before and after the threshold is met as evidence.
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 SFR.

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.

2.2 Cryptographic Support (FCS)

2.2.1 FCS_CKM.1 Cryptographic Key Generation

2.2.1.1 TSS

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

Findings:	[ST] Section 6.2.1 – The TOE supports the following key generation schemes: a) RSA 2048-bit. Used in SSH and TLS RSA ciphersuites. b) ECC P-256/P-384/P-521. Used in TLS. c) FFC safe-prime groups. Diffie-Hellman used in TLS and SSH.

2.2.1.2 Guidance Documentation

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

Findings:	[SUPP] page 15 “Enabling administrative access” Section - The Diffie-Hellman group should be set to Group 14 (2048-bit modulus) as per the evaluated configuration: config system global set dh-params 2048 end The TLS channel for audit offload and SSH trusted path cryptographic characteristics are not modifiable by the user.

2.2.1.3 Tests

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

Key Generation for FIPS PUB 186-4 RSA Schemes

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

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

a. Random Primes:

- Provable primes
- Probable primes

b. Primes with Conditions:

- Primes 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

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

Key Generation for Elliptic Curve Cryptography (ECC)

FIPS 186-4 ECC Key Generation Test

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

FIPS 186-4 Public Key Verification (PKV) Test

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

Key Generation for Finite-Field Cryptography (FFC)

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

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

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

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

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

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

- 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.

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

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

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

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

39 for each FFC parameter set and key pair.

NOTE: The following requirement has been modified by TD0580

FFC Schemes using "safe-prime" groups

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

Findings:	RSA, and ECC key generation is covered by the CAVP certificates listed in Table 4 of the [ST] which claims RSA KeyGen, and ECDSA KeyGen. These claims are consistent with FCS_CKM.1 and FCS_CKM.2 in the [ST] section 5.3.2. The evaluator verified that the crypto modules used in the TOE correspond to the crypto modules which have CAVP-validated cryptographic algorithms.

2.2.2 FCS_CKM.2 Cryptographic Key Establishment

2.2.2.1 TSS

NOTE: The following requirement has been modified by TD0580

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

Findings: [ST] Section 6.2.2 – The TOE supports the following key establishment schemes:
a) ECC schemes. Used in TLS ciphersuites with ECDH key exchange. TOE is both sender and receiver.
b) FFC schemes. Diffie-Hellman used in SSH and TLS ciphersuites with DH key exchange. TOE is both sender and receiver. The TOE supports safe prime DH group 14.

Table 13: Key Agreement Mapping

Scheme	SFR	Service
ECC	FCS_TLSS_EXT.1	Administration
	FCS_TLSC_EXT.1/2	Audit Server
FFC	FCS_SSHS_EXT.1	Administration
	FCS_TLSS_EXT.1	Administration
	FCS_TLSC_EXT.1/2	Audit Server

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

Scheme	SFR	Service
RSA	FCS_TLSS_EXT.1	Administration
ECDH	FCS_SSHC_EXT.1	Audit Server
ECDH	FCS_IPSEC_EXT.1	Authentication Server

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

Findings: [ST] Section 6.2.2 supplies the necessary information.

2.2.2.2 Guidance Documentation

44 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: [SUPP] page 16 “Enabling administrative access” Section – The Diffie-Hellman group should be set to Group 14 (2048-bit modulus) as per the evaluated configuration. TLS

channels to the remote (non-TOE) FortiAnalyzer and SSH trusted path cryptographic characteristics are not modifiable by the user.

2.2.2.3 Tests

NOTE: The following requirement has been modified by TD0580

Key Establishment Schemes

45 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

46 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

47 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.

48 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.

49 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.

50 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.

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

Validity Test

52 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.

53 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).

54 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:	Key establishment schemes are covered by the CAVP certificates listed in Table 4 of the [ST] for KAS-ECC Component, and KAS-FFC Component. These claims are consistent with FCS_CKM.2 in the [ST] section 5.3.2. The evaluator verified that the crypto modules used in the TOE correspond to the crypto modules which have CAVP-validated cryptographic algorithms.

RSA-based key establishment schemes

55 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.

Note	The ST does not claim RSA-based key establishment schemes.

FFC Schemes using "safe-prime" groups

56 The evaluator shall verify the correctness of the TSF's implementation of safe-prime groups by using a known good implementation for each protocol selected in FTP_TRP.1/Admin, FTP_TRP.1/Join, FTP_ITC.1 and FPT_ITT.1 that uses safe-prime groups. This test must be performed for each safe-prime group that each protocol uses.

High-Level Test Description
The test cases for FTP_TRP.1/Admin and FTP_ITC.1 verify the correctness of the FFC schemes using "safe-prime" groups. Those test cases use an independent, known-good interoperable cryptographic implementation. Diffie-Hellman group 14 is shown to work.
Findings: PASS

2.2.3 FCS_CKM.4 Cryptographic Key Destruction

2.2.3.1 TSS

57 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:	[ST] Section 6.2.3 and the table provided in section 6.5.1 in the [ST] lists all keys, the supported generation algorithms, storage medium, and zeroization (destruction).

58 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:	[ST] Section 6.2.3 and the table provided in section 6.5.1 in the [ST] lists how each key gets deleted and interfaces used to delete each key.

59 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:	[ST] Section 6.2.3 and the table provided in section 6.5.1 in the [ST] lists the interfaces used to destroy plaintext keys. Keys stored in volatile memory do not destroy keys via "*destruction of a reference*". However, all selections in FCS_CKM.4.1 fall under "invocation of an interface" and therefore, the evaluator considered each key stored in non-volatile memory and their means of destruction to ensure it was consistent with the claimed interface. Long-term keys stored in non-volatile storage are destroyed using the "erase-disk" command, according to the table in section 6.5.1 of the [ST]. The semantics of the command described in the [CLI] document is consistent with the claims in the [ST].

60 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:	[ST] Section 6.2.3 and the table provided in section 6.5.1 in the [ST] indicates that all private keys stored in non-volatile storage are stored in plaintext.

61 The evaluator shall check that the TSS identifies any configurations or circumstances that may not conform to the key destruction requirement (see further discussion in the Guidance Documentation section below). Note that reference may be made to the Guidance Documentation for description of the detail of such cases where destruction may be prevented or delayed.

[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.

Findings:	[ST] Section 6.2.3 does not indicate any circumstances that may not conform to the key destruction requirements.

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

Findings:	The [ST] does not claim this selection.

2.2.3.2 Guidance Documentation

63 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.

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

Findings:	There are no obvious circumstances where delayed or prevented key destruction can occur. The “Key Zeroization” section in the [SUPP] describes the process for clearing CSPs and other sensitive information from the TOE when required.

2.2.3.3 Tests

65 None

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

2.2.4.1 TSS

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

Findings:	[ST] Section 6.2.4 lists AES key sizes and modes. These algorithms are consistent with the SFR claims.

[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).

2.2.4.2 Guidance Documentation

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

Findings:	These parameters are non-configurable and supported by default.

2.2.4.3 Tests

AES-CBC Known Answer Tests

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

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

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

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

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

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

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

the value that results in an all-zeros plaintext when decrypted with its corresponding key.

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

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

AES-CBC Multi-Block Message Test

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

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

AES-CBC Monte Carlo Tests

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

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

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

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

AES-GCM Test

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

128 bit and 256 bit keys

a. **Two plaintext lengths**. One of the plaintext lengths shall be a non-zero integer multiple of 128 bits, if supported. The other plaintext length shall not be an integer multiple of 128 bits, if supported.

a. **Three AAD lengths**. One AAD length shall be 0, if supported. One AAD length shall be a non-zero integer multiple of 128 bits, if supported. One AAD length shall not be an integer multiple of 128 bits, if supported.

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

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

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

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

AES-CTR Known Answer Tests

86 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. Since the 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):

87 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.

88 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.

89 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.

90 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].

91 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

92 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

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

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

94 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.

95 There is no need to test the decryption engine.

Findings:	AES encryption and decryption is covered by the following CAVP certificates for 128 and 256-bit AES in CBC, GCM and CTR: C2013, A1062, C1908, A2401. These claims are consistent with FCS_COP.1/DataEncryption in the [ST] section 5.3.2. The evaluator verified that the crypto modules used in the TOE correspond to the crypto modules which have CAVP-validated cryptographic algorithms.

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

2.2.5.1 TSS

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

Findings:	[ST] Section 6.2.5 lists the cryptographic algorithms supported by the TOE for signature generation and verification services. These algorithms are consistent with the SFR claims.

2.2.5.2 Guidance Documentation

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

Findings:	[ADMIN] “Certificates” section starting on page 186 details how to set the algorithm and key sizes for CSRs.

2.2.5.3 Tests

ECDSA Algorithm Tests

ECDSA FIPS 186-4 Signature Generation Test

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

ECDSA FIPS 186-4 Signature Verification Test

99 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:	ECDSA signature generation and verification is covered by the following CAVP certificate which claim ECDSA Signature Algorithm with NIST curves P-256, P-384 and P-521: A1062. These claims are consistent with FCS_COP.1/SigGen in the [ST] section 5.3.2. The evaluator verified that the crypto modules used in the TOE correspond to the crypto modules which have CAVP-validated cryptographic algorithms.

RSA Signature Algorithm Tests

Signature Generation Test

100 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.

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

Signature Verification Test

102 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.

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

Findings:	RSA signature generation and verification is covered by the following CAVP certificate which claim RSA Signature Algorithm with key size of 2048: A1062 and C2013. These claims are consistent with FCS_COP.1/SigGen in the [ST] section 5.3.2. The evaluator verified that the crypto modules used in the TOE correspond to the crypto modules which have CAVP-validated cryptographic algorithms.

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

2.2.6.1 TSS

104 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:	[ST] Section 6.2.6 lists the parts of the TSF that implement the claimed hashing algorithms. These algorithms are consistent with the SFR claims.

2.2.6.2 Guidance Documentation

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

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

2.2.6.3 Tests

106 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.

107 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

108 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

109 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

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

Selected Long Messages Test - Byte-oriented Mode

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

Pseudorandomly Generated Messages Test

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

Findings:	Hashing is covered by the following CAVP certificates which claim SHA-1, SHA-256, SHA-384, SHA-512: C2013 and A1062. These claims are consistent with FCS_COP.1/Hash in the [ST] section 5.3.2. The evaluator verified that the crypto modules used in the TOE correspond to the crypto modules which have CAVP-validated cryptographic algorithms.

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

2.2.7.1 TSS

113 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:	[ST] Section 6.2.7 and the table in section 6.2.7 of the [ST] lists the HMAC algorithms used. These algorithms are consistent with the SFR claims. The detail includes the block size, key size, and output digest size. The hash function used is directly implied by the HMAC algorithm.

2.2.7.2 Guidance Documentation

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

Findings:	No manual configuration is necessary, keyed-hash algorithm configuration is enabled by default.

2.2.7.3 Tests

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

Findings:	HMAC is covered by the following CAVP certificates which claim HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-384 and HMAC-SHA-512: C2013 and A1062. These claims are consistent with FCS_COP.1/KeyedHash in the [ST] section 5.3.2 and Table 14: HMAC Characteristics. The evaluator verified that the crypto modules used in the TOE correspond to the crypto modules which have CAVP-validated cryptographic algorithms.

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

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

2.2.8.1 TSS

117 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:	[ST] Section 6.2.9 indicates that the TOE uses a CTR_DRBG which is permitted under SP 800-90A. The DRBG is seeded from a hardware-based entropy generator known as the “Fortinet Entropy Token”. The entropy is derived from wide-band radio-frequency white noise. The entropy seed contains at least 256-bits entropy. More details are discussed in the evaluated confidential entropy description document.

2.2.8.2 Guidance Documentation

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

Findings:	[SUPP] Page 9 “Entropy” Section has several sections including installing the token, configuring the entropy token settings and setting RBG reseed interval. Also [CLI] page 64 “fips” section shows the command for setting the entropy-token to enable/disable.

2.2.8.3 Tests

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

120 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).

121 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.

122 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:	CTR_DRBG is covered by the following CAVP certificates which claims the use of a 256-bit key: C1985 and A1962. These claims are consistent with FCS_RBG_EXT.1 in the [ST] section 5.3.2. The evaluator verified that the crypto modules used in the TOE correspond to the crypto modules which have CAVP-validated cryptographic algorithms.

2.3 Identification and Authentication (FIA)

2.3.1 FIA_AFL.1 Authentication Failure Management

2.3.1.1 TSS

123 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:	[ST] Section 6.3.5 describes that when a user fails authentication the configured number of times, they are locked out for an administrator configured amount of time.

124 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:	[ST] Section 6.3.5 states that the local console does not implement the lockout mechanism which means there is no instance in which the administrator can lose control of the TOE.

2.3.1.2 Guidance Documentation

125 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:	[ADMIN] “Password lockout and retry attempts” section explains how to configure the number of unsuccessful login attempts and the lockout duration.

126 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:	No actions are required to ensure admin access is always maintained. By default the local console does not lockout the administrator.

2.3.1.3 Tests

127 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. Attempt to login over SSH again. The attempt should fail. Wait 80 seconds and attempt to login. The attempt should succeed. Repeat the above test using the Web GUI interface instead of the SSH interface. Repeat the above test with a lockout duration of 3 minutes and 5 minutes.
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).

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

High-Level Test Description
The TOE used time based lockouts; these are tested in the previous test case.
Findings: PASS

2.3.2 FIA_PMG_EXT.1 Password Management

2.3.2.1 TSS

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

Findings:	[ST] Section 6.3.1 – Passwords can be composed of a character set consistent with the SFR. The minimum password length is administrator configurable between 6 and 128 characters.

2.3.2.2 Guidance Documentation

129 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:	[SUPP] Page 11 “The FIPS-CC Mode of Operation” section shows that in FIPS/CC mode the password must be at least 8 characters and must contain an upper-case letter, lower-case letter, numeral and non-alphanumeric character. [CLI] page 98 “password-policy” section shows how to change the passwords minimum length from the CLI. [ADMIN] page 246 “Password policy” section also contains this information and shows how to change the minimum length from the GUI.

2.3.2.3 Tests

130 The evaluator shall perform the following tests.

a. Test 1: The evaluator shall compose passwords that 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

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

High-Level Test Description
See previous test case.
Findings: PASS

2.3.3 FIA_UIA_EXT.1 User Identification and Authentication

2.3.3.1 TSS

131 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:	[ST] Section 6.3.2 and [ST] Section 6.3.3 – Lists the available interfaces for a user to login to and the logon processes for them and what constitutes a “successful logon”. The description of the login process is consistent with the norms of the specified interfaces.

132 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:	[ST] Section 6.3.2 and [ST] Section 6.3.3 - The TOE warning banner can be viewed prior to authentication. No TOE administrative access is permitted until an administrator is successfully identified and authenticated.

133 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.

134 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.

2.3.3.2 Guidance Documentation

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

Findings:	[SUPP] Page 15 “Enabling administrative access” shows how to enable the various methods to login. [SUPP] Page 16 “Admin access disclaimer” section and [CLI] page 66 “global” section show how to enable the pre-login-banner. [CLI] Page 46 “admin user” section shows how to set ssh public keys for a user. [ADMIN] Page 10 “Connecting to the GUI” section explains how to login to the GUI. [CLI] Page 21 “Connecting to the FortiAnalyzer console” has details on how to connect to the CLI.

2.3.3.3 Tests

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

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

High-Level Test Description
Log into the identified management interface using a known-good credential and logout. Attempt to login into the identified management interface using a known-bad credential and verify that the attempt fails. 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. All claimed services available to remote entities are identified as part of AVA_VAN.1 test scanning.
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 identification and authorization. All claimed services available to local entities are identified as part of AVA_VAN.1 test scanning.
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.

Test Not Applicable	The TOE is not a distributed TOE

2.3.4 FIA_UAU_EXT.2 Password-based Authentication Mechanism

137 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.

2.3.5 FIA_UAU.7 Protected Authentication Feedback

2.3.5.1 TSS

138 None

2.3.5.2 Guidance Documentation

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

Findings:	The TOE does not reveal authentication data by default when in FIPS/CC mode.

2.3.5.3 Tests

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

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

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

2.4 Security management (FMT)

2.4.1 General requirements for distributed TOEs

2.4.1.1 TSS

141 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.

2.4.1.2 Guidance Documentation

142 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.

2.4.1.3 Tests

143 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).

Test Not Applicable	The TOE is not a distributed TOE.

2.4.2 FMT_MOF.1/ManualUpdate

2.4.2.1 TSS

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

Findings:	The TOE is not a distributed TOE.

2.4.2.2 Guidance Documentation

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

Findings:	[SUPP] Page 7 "Downloading the FIPS-CC certified firmware", "Verifying the integrity of the firmware build", and "Installing the FIPS-CC firmware build" sections detail how to install the firmware on the TOE. [ADMIN] Page 160 "Updating the system firmware" also details how to update the firmware from the GUI. The [AGD] does not define any functions which cease to operate during an update as there are none that are applicable.

146 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.

2.4.2.3 Tests

147 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.

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

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

2.4.3 FMT_MTD.1/CoreData Management of TSF Data

2.4.3.1 TSS

149 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:	[ST] Section 6.4.3 - Users are required to login before being provided with access to any administrative function.

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

Findings:	[ST] Section 6.4.3 - Management of TSF data using the CLI or web GUI is restricted to Security Administrators.

2.4.3.2 Guidance Documentation

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

Findings:	The various TSF data manipulating functions are found throughout the [CLI], [ADMIN], and [SUPP] documents. For specific references please see the related SFRs.

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

Findings:	[SUPP] Page 13 "Install CA Certificate" and "Install Updated Certificates" sections detail how to import ca certificates. [CLI] Page 59 Section “certificate” also details all the various commands for loading various types of certificates. [ADMIN] Page 186 “Certificates” section also details the various certificates that can be loaded into the TOE and how to do that.

2.4.3.3 Tests

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

2.4.4 FMT_SMF.1 Specification of Management Functions

154 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.

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

155 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:	[ST] Section 6.4.6 – Lists management functions that are available in the TOE. All management functions are available in all interfaces.

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

Findings:	[ST] Section 6.3.3 indicates that the TOE offers a direct connection to the TOE appliance. The quick start guides [QSG] provided for each of the claimed hardware models in the [ST] section 2.4 explicitly indicate console RS-232 over RJ-45 ports (FAZ-800F), and DB9 (FAZ-1000F, FAZ-2000E, FAZ-3000F, FAZ-3500G, FAZ-3700F) used to access the direct serial console. Given these local console mechanisms are distinct physical interfaces, no warnings are appropriate or necessary (i.e., this is not an instance where there are multiple RJ-45 ethernet-style adaptors in which an ethernet cross-over cable is used to provide direct line-of-sight access to the device).

157 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.

2.4.4.2 Guidance Documentation

158 See section 2.4.4.1.

2.4.4.3 Tests

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

2.4.5 FMT_SMR.2 Restrictions on security roles

2.4.5.1 TSS

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

Findings:	[ST] Section 6.4.4 – Lists the types of user profiles and the restrictions for each. The “Super_User” role equates to the “Security Administrator” role in the Protection Profile. Management of TSF data is restricted to Security Administrators.

2.4.5.2 Guidance Documentation

161 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:	[SUPP] “Enabling administrative access” section describes how to enable remote administrative access to the TOE. [SUPP] Page 15 “Remote access requirements” outlines the requirements the admin must meet in order to remotely access the TOE. The [QSG] specifies the use of the console port to access the CLI locally.

2.4.5.3 Tests

162 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.

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

2.5 Protection of the TSF (FPT)

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

2.5.1.1 TSS

163 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:	[ST] Section 6.5.1 – Lists types of keys and how each are stored. The section also states the TOE prevents the reading of all pre-shared keys, symmetric keys and private keys stored in the TOE. The table provided in section 6.5.1 of the [ST] indicates all keys are stored in plaintext.

2.5.2 FPT_APW_EXT.1 Protection of Administrator Passwords

2.5.2.1 TSS

164 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:	[ST] Section 6.5.2 – Passwords are hashed using SHA-256. Administrator passwords are stored in non-volatile storage and are overwritten with zeroes by erase-disk command.

2.5.3 FPT_TST_EXT.1 TSF testing

2.5.3.1 TSS

165 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:	This information can be found in section 6.5.3 of the [ST]. The TOE performs a typical boot process. The TOE verifies its kernel, firmware and software images using 2048-bit RSA signature and executes cryptographic known answer tests (KATs). Section 6.5.3 of the [ST] provides an argument that the tests are sufficient to demonstrate that the TSF is operating correctly.

166 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.

2.5.3.2 Guidance Documentation

167 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:	[SUPP] Page 17 “FIPS Error Mode” section details what happens if a test fails and what steps the administrator should take.

168 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.

2.5.3.3 Tests

169 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.

170 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.

171 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.

172 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

2.5.4 FPT_TUD_EXT.1 Trusted Update

2.5.4.1 TSS

173 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:	[ST] Section 6.5.4 - The current firmware version may be queried using either the CLI or the UI using the means documented in the [SUPP]. The [ST] does not claim delayed activation.

174 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:	[ST] Section 6.5.4 indicates that end-users can download new software/firmware from Fortinet's support website. The TOE uses a digital signature to verify the update and if the digital signature cannot be verified the update fails.

175 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:	Both 'support automatic checking for updates' and 'support automatic updates' were not chosen for the selection in FPT_TUD_EXT.1.2.

176 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.

177 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:	A published hash is not used to verify updates.

2.5.4.2 Guidance Documentation

178 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:	[SUPP] Page 7 “Installing the FIPS-CC firmware build” shows how to query the currently active version. [CLI] Page 251 “system status” shows the CLI command to get the version. [ADMIN] Page 156 “Dashboard” section shows where the version can be found in the web GUI. Delayed activation is not supported.

179 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:	[SUPP] Page 7 “Verifying the integrity of the firmware build” section - In addition to the above, FIPS validated firmware images are digitally signed via an RSA key. Any FIPS validated firmware image that is installed or updated will automatically be verified with this signature.

180 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:	Published hashes are not claimed.

181 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.

182 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.

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

Findings:	Certificate based update authentication is not claimed.

2.5.4.3 Tests

184 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. Install a legitimate same-grade version of the TOE. After the install, get the current version of the TOE and ensure it is consistent.
Findings: PASS

b. Test 2 [conditional]: If the TOE itself verifies a digital signature to authorize the installation of an image to update the TOE the following test shall be performed (otherwise the test shall be omitted). 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
Attempt to upgrade the TOE with an image that is modified, an image that is unsigned, and an image that contains an invalid signature. Show that all of these upgrade attempts fail.
Findings: PASS

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

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

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

3) If the TOE allows delayed activation of updates, the TOE must be able to display both the currently executing version and most recently installed version. The handling of version information of the most

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

Test Not Applicable	The TOE does not support published hashes.

185 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.

186 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.

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

Test Not Applicable	The TOE is not a distributed TOE.

2.5.5 FPT_STM_EXT.1 Reliable Time Stamps

2.5.5.1 TSS

188 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:	[ST] Section 6.5.5 –The time is set by the administrator. The TOE uses the time in the audit record timestamps, session timeouts, and certificate validation. The TOE incorporates an internal clock which can be considered reliable.

189 **TD0632 -** If “obtain time from the underlying virtualization system” is selected, the evaluator shall examine the TSS to ensure that it identifies the VS interface the TOE uses to obtain time. If there is a delay between updates to the time on the VS and updating the time on the TOE, the TSS shall identify the maximum possible delay.

Findings:	**“**obtain time from the underlying virtualization system” is not selected.

2.5.5.2 Guidance Documentation

190 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:	[CLI] Page 150 “date” section shows how to set the date and time in the CLI. NTP is not claimed and must be disabled as per [SUPP] Page 17 “Disable NTP” section.

191 **TD0632 -**If the TOE supports obtaining time from the underlying VS, the evaluator shall verify the Guidance Documentation specifies any configuration steps necessary. If no configuration is necessary, no statement is necessary in the Guidance Documentation. If there is a delay between updates to the time on the VS and updating the time on the TOE, the evaluator shall ensure the Guidance Documentation informs the administrator of the maximum possible delay.

Findings:	The TOE does not support obtaining time from the underlying VS.

2.5.5.3 Tests

192 The evaluator shall perform the following tests:

a. Test 1: **TD0632 -** 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 to one in the past. Then verify the date/time was set properly. Change the date/time to a time in the future. Then verify the date/time was set properly.
Findings: PASS

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

Test Not Applicable	The TOE does not claim NTP.

193 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 claim NTP.

194 **TD0632 –** c) Test 3: [conditional] If the TOE obtains time from the underlying VS, the evaluator shall record the time on the TOE, modify the time on the underlying VS, and verify the modified time is reflected by the TOE. If there is a delay between the setting the time on the VS and when the time is reflected on the TOE, the evaluator shall ensure this delay is consistent with the TSS and Guidance.

Test Not Applicable	The TOE does not claim obtaining time from an underlying VS.

2.6 TOE Access (FTA)

2.6.1 FTA_SSL_EXT.1 TSF-initiated Session Locking

2.6.1.1 TSS

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

Findings:	[ST] Section 6.6.1 - The Security Administrator may configure the TOE to terminate an inactive local administrative session following a specified period of time. The timeout value is set to five minutes by default but is configurable.

2.6.1.2 Guidance Documentation

196 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:	[ADMIN] page 248 “idle timeout” section shows how to change the idle timeout period from the GUI. [CLI] Page 43 “admin setting” section describes how to change the idle timeout from the CLI.

2.6.1.3 Tests

197 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; Log into the device;

High-Level Test Description
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

2.6.2 FTA_SSL.3 TSF-initiated Termination

2.6.2.1 TSS

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

Findings:	[ST] Section 6.6.2 - The Security Administrator may configure the TOE to terminate an inactive remote CLI or Web GUI session following a specified period of time. The timeout value is set to five minutes by default but is configurable.

2.6.2.2 Guidance Documentation

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

Findings:	[ADMIN] page 248 “Idle timeout” section shows how to change the idle timeout period from the GUI. [CLI] Page 43 “admin setting” section describes how to change the idle timeout from the CLI.

2.6.2.3 Tests

200 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 it in one interface and check that it is enforced in all remote administration interfaces.
Findings: PASS

2.6.3 FTA_SSL.4 User-initiated Termination

2.6.3.1 TSS

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

Findings:	[ST] Section 6.6.3 - Administrative users may terminate their own sessions at any time.

2.6.3.2 Guidance Documentation

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

Findings:	[SUPP] Page 16 “Logging out from the GUI and CLI” section contains instructions on terminating CLI and GUI connections.

2.6.3.3 Tests

203 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 previously 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 previously discussed.
Verify that the session has been terminated.
Log into the Web interface.
Copy the URL presented.
Log out using the TSFI previously discussed.
Paste the URL back into the web browser and attempt to navigate to it.
Findings: PASS

2.6.4 FTA_TAB.1 Default TOE Access Banners

2.6.4.1 TSS

204 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:	[ST] Section 6.6.4 - TOE Administrators may access the TOE remotely (via the HTTPS/TLS web GUI or SSH) or locally (via the serial/console port). The TOE displays an administrator configurable message to users prior to authentication.

2.6.4.2 Guidance Documentation

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

Findings:	[SUPP] Page 16 “Admin access disclaimer” section shows how to enable/disable the banner. [CLI] Page 66 “global” section shows the CLI command to enable/disable the pre login banner and the command to change the message.

2.6.4.3 Tests

206 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.
Findings: PASS

2.7 Trusted path/channels (FTP)

2.7.1 FTP_ITC.1 Inter-TSF trusted channel

2.7.1.1 TSS

207 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:	[ST] Section 6.7.1 – The TOE interfaces with another remote FortiAnalyzer device to offload audit records (as per [ST] section 6.1.3). This "audit server" channel is claimed against FCS_TLSC_EXT.1 and FCS_TLSC_EXT.2 and section 6.2.11 of the [ST] claims this is a TLS client connection. Furthermore, sections 6.2.11 and 6.2.12 of the [ST] indicate that the connection is mutually authenticated using X.509 certificates.

2.7.1.2 Guidance Documentation

208 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:	[SUPP] Page 19 "FortiAnalyzer configuration" section details how to establish a connection with an external FortiAnalyzer device. [SUPP] Page 19 "Reconnecting to a remote FortiAnalyzer unit" section contains information on reconnecting to an external FortiAnalyzer device if the connection is interrupted.

2.7.1.3 Tests

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

210 The evaluator shall perform the following tests:

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

Note	The TOE maintains one trusted channel in the log forwarding channel allowing the TOE to send logs to another FortiAnalyzer device over TLS. This channel 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.

Findings:	Refer to previous test

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

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

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

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

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 steps with an 8 minute timeout performing a login attempt every 30 seconds.
Findings: PASS

Further assurance activities are associated with the specific protocols.

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

Findings:	This is not a distributed TOE.

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

2.7.2 FTP_TRP.1/Admin Trusted Path

2.7.2.1 TSS

213 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:	[ST] Section 6.7.2 - The TOE provides CLI connection over SSH per FCS_SSHS_EXT.1 and Web GUI over HTTPS per FCS_HTTPS_EXT.1.1 for remote administration. The protocols are consistent with the claims in the [ST] section 5.3.2.

2.7.2.2 Guidance Documentation

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

Findings:	[ADMIN] Page 10 “Connecting to the GUI” section outlines how to establish a session with the TOEs web GUI. [CLI] Page 20 “Using the Command Line Interface” section contains information on how to enable the command line and how to use it.

2.7.2.3 Tests

215 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.

High-Level Test Description
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

216 Further assurance activities are associated with the specific protocols.

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

Findings:	The TOE is not a distributed TOE.

3 Evaluation Activities for Optional Requirements

3.1.1 FCS_TLSC_EXT.2 Extended: TLS Client support for mutual authentication

3.1.1.1 TSS

FCS_TLSC_EXT.2.1

218 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:	[ST] Section 6.2.12 – The TOE supports presentation of an X.509v3 client certificate for authentication as required by the FortiAnalyzer Audit Server.

3.1.1.2 Guidance Documentation

FCS_TLSC_EXT.2.1

219 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:	Steps for loading this certificate can be found in [CLI] page 61 “certificate oftp” section.

3.1.1.3 Tests

FCS_TLSC_EXT.2.1

220 **TD0670 -** For all tests in this chapter the TLS server used for testing of the TOE shall be configured to require mutual authentication.

221 FCS_TLSC_EXT.2.1

222 Test 1: The evaluator shall establish a connection to a peer server that is configured for mutual authentication (i.e. sends a server Certificate Request (type 13) message). The evaluator observes that the TOE TLS client sends both client Certificate (type 11) and client Certificate Verify (type 15) messages during its negotiation of a TLS channel and that Application Data is sent.

223 In addition, all other testing in FCS_TLSC_EXT.1 and FIA_X509_EXT.* must be performed as per the requirements.

Note:	The TOE was tested under FCS_TLSC_EXT.1.1 and connections were verified to contain a server Certificate Request, and client messages of Certificate and Certificate Verify types. The connections were completed successfully and application data was sent.

4 Evaluation Activities for Selection-Based Requirements

4.1 Cryptographic Support (FCS)

4.1.1 FCS_HTTPS_EXT.1 HTTPS Protocol

4.1.1.1 TSS

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

Findings:	[ST] Section 6.2.8 – The TOE is RFC 2818 compliant and the evaluator deems the explanation sufficient after reviewing the RFC.

4.1.1.2 Guidance Documentation

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

Findings:	[SUPP] Page 15 “Enabling administrative access” section shows how to enable https. [ADMIN] Page 10 “Connecting to the GUI” details how to connect and use the GUI. [SUPP] Page 15 “Web browser requirements” section details the cryptographic parameters.

4.1.1.3 Tests

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

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

228 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.

4.1.2 FCS_SSHS_EXT.1 SSH Server

4.1.2.1 TSS

FCS_SSHS_EXT.1.2

229 **TD0631 -** The evaluator shall check to ensure that the TSS contains a list of supported public key algorithms that are accepted for client authentication and that this list is consistent with signature verification algorithms selected in FCS_COP.1/SigGen (e.g., accepting EC keys requires corresponding Elliptic Curve Digital Signature algorithm claims).

230 **TD0631 -** The evaluator shall confirm that the TSS includes the description of how the TOE establishes a user identity when an SSH client presents a public key or X.509v3 certificate. For example, the TOE could verify that the SSH client’s presented public key matches one that is stored within the SSH server’s authorized_keys file.

231 **TD0631 -** If password-based authentication method has been selected in the FCS_SSHS_EXT.1.2, then the evaluator shall confirm its role in the authentication process is described in the TSS.

Findings:	[ST] Section 6.2.10 - The TOE supports password-based or public key (ssh-rsa, rsa-sha2-256 and rsa-sha2-512) authentication. The password authentication method is explained further in [ST] section 6.3.3.

FCS_SSHS_EXT.1.3

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

Findings:	[ST] Section 6.2.10 - The TOE examines the size of each received SSH packet. If the packet is greater than 256KB, it is automatically dropped.

FCS_SSHS_EXT.1.4

233 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:	[ST] Section 6.2.10 - The TOE supports AES-CTR-128 and AES-CTR-256 encryption algorithms. These encryption algorithms match those listed for this component. No optional characteristics are specified or expected.

FCS_SSHS_EXT.1.5

234 **TD0631 -** The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that the SSH server’s host public key algorithms supported are specified and that they are identical to those listed for this component.

Findings:	[ST] Section 6.2.10 – The TOE uses ssh-rsa, rsa-sha2-256 and rsa-sha2-512 as its public key algorithms which match those listed for this component. No optional characteristics are specified or expected.

235 The evaluator shall confirm that the TSS includes the description of how the TOE establishes a user identity when an SSH client presents a public key or X.509v3 certificate. For example, the TOE could verify that the SSH client’s presented public key matches one that is stored within the SSH server’s authorized_keys file.

Findings:	This information can be found in section 6.2.10 of the [ST] which claims that the TOE stores the public key so that it can be verified against the user’s signed authentication message.

FCS_SSHS_EXT.1.6

236 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:	[ST] Section 6.2.10 – The TOE supports HMAC-SHA1, HMAC-SHA2-256 and HMAC-SHA2-512 data integrity algorithms which matches the list in this component.

FCS_SSHS_EXT.1.7

237 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:	[ST] Section 6.2.10 – The TOE supports diffie-hellman-group14-sha1 for SSH key exchanges which matches the list in this component.

FCS_SSHS_EXT.1.8

238 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.

Findings:	[ST] Section 6.2.10 - The TOE will re-key SSH connections after 1 hour or after an aggregate of 1 gigabyte of data has been exchanged (whichever occurs first).

4.1.2.2 Guidance Documentation

FCS_SSHS_EXT.1.4

239 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:	No configuration is needed to ensure SSH conforms to the description in the TSS.

FCS_SSHS_EXT.1.5

240 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:	No configuration is needed to ensure SSH conforms to the description in the TSS

FCS_SSHS_EXT.1.6

241 The evaluator shall also check the guidance documentation to ensure that it contains instructions to the Security 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:	No configuration is needed to ensure SSH conforms to the description in the TSS

FCS_SSHS_EXT.1.7

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

Findings:	No configuration is needed to ensure SSH conforms to the description in the TSS

FCS_SSHS_EXT.1.8

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

the guidance documentation describes that the TOE reacts to the first threshold reached.

Findings:	These thresholds are not configurable.

4.1.2.3 Tests

FCS_SSHS_EXT.1.2

244 **TD0631 -** Test objective: The purpose of these tests is to verify server supports each claimed client authentication method.

245 **TD0631 -** Test 1: For each supported client public-key authentication algorithm, the evaluator shall configure a remote client to present a public key corresponding to that authentication method (e.g., 2048-bit RSA key when using ssh-rsa public key). The evaluator shall establish sufficient separate SSH connections with an appropriately configured remote non-TOE SSH client to demonstrate the use of all applicable public key algorithms. It is sufficient to observe the successful completion of the SSH Authentication Protocol to satisfy the intent of this test.

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

246 **TD0631 -** Test 2: The evaluator shall choose one client public key authentication algorithm supported by the TOE. The evaluator shall generate a new client key pair for that supported algorithm without configuring the TOE to recognize the associated 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
Using an SSH client, connect to the TOE server using a private key that does not match the public key half loaded in the TOE. The TOE will reject the authentication attempt.
Findings: PASS

247 **TD0631 -** Test 3: [Conditional] If password-based authentication method has been selected in the FCS_SSHS_EXT.1.2, 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 connecting SSH client.

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

248 **TD0631 -** Test 4: [Conditional] If password-based authentication method has been selected in the FCS_SSHS_EXT.1.2, the evaluator shall configure the TOE to accept password-based authentication and demonstrate that user authentication fails when the incorrect password is provided by the connecting SSH client.

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

FCS_SSHS_EXT.1.3

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

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

FCS_SSHS_EXT.1.4

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

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

FCS_SSHS_EXT.1.5

251 **TD0631 -** Test objective: This test case is meant to validate that the TOE server will support host public keys of the claimed algorithm types.

252 **TD0631 -** Test 1: The evaluator shall configure (only if required by the TOE) the TOE to use each of the claimed host public key algorithms. The evaluator will then use an SSH client to confirm that the client can authenticate the TOE server public key using the claimed algorithm. 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
Use an SSH client to confirm that the client can authenticate the TOE host public key using the claimed host public key algorithm.
Findings: PASS

253 **TD0631 -** Test objective: This negative test case is meant to validate that the TOE server does not support host public key algorithms that are not claimed.

254 **TD0631 -** Test 2: The evaluator shall configure a non-TOE SSH client to only allow it to authenticate an SSH server host public key algorithm that is not included in the ST selection. The evaluator shall attempt to establish an SSH connection from the non-TOE SSH client to the TOE SSH server and observe that the connection is rejected.

High-Level Test Description
Use an SSH client to confirm that the connection is rejected when the client attempts to authenticate the TOE via a host public key algorithm that is not claimed in the ST.
Findings: PASS

FCS_SSHS_EXT.1.6

255 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.

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

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

257 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.

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

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

FCS_SSHS_EXT.1.7

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

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

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

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

FCS_SSHS_EXT.1.8

261 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.

262 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).

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

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

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

265 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).

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

High-Level Test Description
Using a custom SSH client, connect to the TOE and send large amounts of data over the channel. Ensure that the TOE rekeys before 500 MB in the aggregate has been transmitted. Ensure that the TOE is responsible for sending the rekey initiation.
Findings: PASS

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

Findings:	These limits are not configurable for this TOE.

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

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

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

Findings:	The TOE does not have hardware limitations.

4.1.3 FCS_TLSC_EXT.1 Extended: TLS Client Protocol without mutual authentication

4.1.3.1 TSS

FCS_TLSC_EXT.1.1

269 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:	[ST] Section 6.2.11 lists the supported ciphersuites. These match those listed in section 5.3.2 of the [ST] for this component.

FCS_TLSC_EXT.1.2

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

Findings:	[ST] Section 6.2.11 – When the TLS client receives an X.509 certificate from the server, the client will compare the reference identifier with the established Subject Alternative Names. The TOE supports wildcards for DNS names in the CN and SAN of the X.509 certificate.

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

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

Findings:	The TOE is not a distributed TOE.

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

Findings:	In [ST] section 6.2.11, the TOE does not support IP addresses in the CN.

FCS_TLSC_EXT.1.4

273 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:	[ST] Section 6.2.11 - The TOE's TLS client will transmit the Supported Elliptic Curves extension in the Client Hello message by default with support for the following curves/groups: secp256r1, secp384r1, and secp521r1.

4.1.3.2 Guidance Documentation

FCS_TLSC_EXT.1.1

274 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 necessary to ensure the TLS client conforms to the description in the TSS.

FCS_TLSC_EXT.1.2

275 The evaluator shall ensure that the operational guidance describes all supported identifiers, explicitly states whether the TOE supports the SAN extension or not and includes detailed instructions on how to configure the reference identifier(s) used to check the identity of peer(s). If the identifier scheme implemented by the TOE includes support for IP addresses, the evaluator shall ensure that the operational guidance provides a set of warnings and/or CA policy recommendations that would result in secure TOE use.

Findings:	[SUPP] Page 19 "FortiAnalyzer configuration" section outlines that SAN and CN are both supported and that IP and DNS are both supported; however, if using an IP address, then the SAN must be present.

276 Where the secure channel is being used between components of a distributed TOE for FPT_ITT.1, the SFR selects attributes from RFC 5280, and FCO_CPC_EXT.1.2 selects “no channel”; the evaluator shall verify the guidance provides instructions for establishing unique reference identifiers based on RFC5280 attributes.

Findings:	The TOE is not a distributed TOE.

FCS_TLSC_EXT.1.4

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

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

4.1.3.3 Tests

FCS_TLSC_EXT.1.1

278 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 all specifically claimed ciphersuites.
Findings: PASS

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

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

280 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 ciphersuite.
Findings: PASS

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

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

High-Level Test Description
Using a Lightship developed 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

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

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

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

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

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

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

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.

High-Level Test Description
Findings: PASS

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

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

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

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

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

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

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 message after the ChangeCipherSpec and verify no application data is sent.
Findings: PASS

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

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. Do this once for a non-DHE ciphersuite and once for a DHE or ECDHE key exchange ciphersuite.
Findings: PASS

FCS_TLSC_EXT.1.2

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

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

or

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

or

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

285 Note that for some tests additional conditions apply.

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

- IPv4: The CN contains a single address that is represented a 32-bit numeric address (IPv4) is written in decimal as four numbers that range from 0-255 separated by periods as specified in RFC 3986.
- IPv6: The CN contains a single IPv6 address that is represented as eight colon separated groups of four lowercase hexadecimal digits, each group representing 16 bits as specified in RFC 4291. Note: Shortened addresses, suppressed zeros, and embedded IPv4 addresses are not tested.

Note	The TOE does not provide for, nor claim, any administrator-defined override mechanism for validating that the reference identifier matches that on the certificates for claimed TLS channels. Therefore, all of the following tests are applicable in the context of FCS_TLSC_EXT.1.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

f. **TD0634 -** Objective: The objective of this test is to ensure the TOE is able to differentiate between IP address identifiers that are not allowed to contain wildcards and other types of identifiers that may contain wildcards.

TD0634 – Test 6 [conditional]: If IP address identifiers supported in the SAN or CN, the evaluator shall present a server certificate that contains a CN that matches the reference identifier, except one of the groups has been replaced with a wildcard asterisk (*) (e.g. CN=*.168.0.1 when connecting to 192.168.0.1, CN=2001:0DB8:0000:0000:0008:0800:200C:* when connecting to 2001:0DB8:0000:0000:0008:0800:200C:417A). The certificate shall not contain the SAN extension. The evaluator shall verify that the connection fails. The evaluator shall repeat this test for each supported IP address version (e.g. IPv4, IPv6).

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

Note	IP Addresses are not supported in the CN.

g. Test 7 [conditional]: If the secure channel is used for FPT_ITT, and RFC 5280 is selected, the evaluator shall perform the following tests. Note, when multiple attribute types are selected in the SFR (e.g. when multiple attribute types are combined to form the unique identifier), the evaluator modifies each attribute type

in accordance with the matching criteria described in the TSS (e.g. creating a mismatch of one attribute type at a time while other attribute types contain values that will match a portion of the reference identifier):

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

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

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

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

Findings:	The TOE does not claim FPT_ITT.1 with RFC 5280.

FCS_TLSC_EXT.1.3

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

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

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

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

Note	This test case is performed as part of FIA_X509_EXT.1. Appropriate override mechanisms are verified.

291 Test 3[conditional]: The purpose of this test to verify that only selected certificate validation failures could be administratively overridden. If any override mechanism is defined for failed certificate validation, the evaluator shall configure a new presented certificate that does not contain a valid entry in one of the mandatory fields or parameters (e.g. inappropriate value in extendedKeyUsage field) but is otherwise valid and signed by a trusted CA. The evaluator shall confirm that the certificate validation fails (i.e. certificate is rejected), and there is no administrative override available to accept such certificate.

Note	This test case is performed as part of FIA_X509_EXT.1. Appropriate override mechanisms are verified.

FCS_TLSC_EXT.1.4

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

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

4.1.4 FCS_TLSS_EXT.1 Extended: TLS Server Protocol

4.1.4.1 TSS

FCS_TLSS_EXT.1.1

293 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:	[ST] Section 6.2.13 – Lists the supported ciphersuites which are identical to those listed in [ST] section 5.3.2 for this component.

FCS_TLSS_EXT.1.2

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

Findings:	[ST] Section 6.2.13 - The server only allows TLS protocol versions 1.1 and 1.2 (rejecting any other protocol version, including SSL 2.0, SSL 3.0 and TLS 1.0 and any other unknown TLS version string supplied).

FCS_TLSS_EXT.1.3

295 **TD0635 -** If using ECDHE and/or DHE ciphers, the evaluator shall verify that the TSS lists all EC Diffie-Hellman curves and/or Diffie-Hellman groups used in the key establishment by the TOE when acting as a TLS Server. For example, if the TOE supports TLS_DHE_RSA_WITH_AES_128_CBC_SHA cipher and Diffie-Hellman parameters with size 2048 bits, then list Diffie-Hellman Group 14.

Findings:	[ST] Section 6.2.13 - The DHE key agreement parameters are restricted to 2048 bits and are hardcoded into the server. ECDHE key agreement parameters are restricted to ECDHE curves secp384r1, secp256r1 and secp521r1. Section 6.2.2 also lists the supported DH Group 14 explicitly.

FCS_TLSS_EXT.1.4

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

297 **TD0569 -** If the TOE claims a (D)TLS server capable of session resumption (as a single context, or across multiple contexts), the evaluator verifies that the TSS describes how session resumption operates (i.e. what would trigger a full handshake, e.g. checking session status, checking Session ID, etc.). If multiple contexts are used the TSS describes how session resumption is coordinated across those contexts. In case session establishment and session resumption are always using a separate context, the TSS shall describe how the contexts interact with respect to session resumption (in particular regarding the session ID). It is acceptable for sessions established in one context to be resumable in another context.

Findings:	[ST] Section 6.2.13 - The TOE supports session resumption based on session IDs according to RFC4346 (TLS1.1) or RFC5246 (TLS1.2), session resumption based on session tickets according to RFC 5077.

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

Findings:	[ST] Section 6.2.13 - Session tickets adhere to the structural format provided in section 4 of RFC 5077.

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

Findings:	[ST] Section 6.2.13 - Session tickets adhere to the structural format provided in section 4 of RFC 5077.

4.1.4.2 Guidance Documentation

FCS_TLSS_EXT.1.1

300 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:	No further configuration is needed to ensure the TLS server conforms with the TSS.

FCS_TLSS_EXT.1.2

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

Findings:	No further configuration is needed to ensure the TLS server conforms with the TSS.

FCS_TLSS_EXT.1.3

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

Findings:	No further configuration is needed to ensure the TLS server conforms with the TSS.

FCS_TLSS_EXT.1.4

303 **NIAP TD0569:** The evaluator shall verify that any configuration necessary to meet the requirement must be contained in the AGD guidance.

Findings:	The TOE supports session resumption based on session tickets by default. No configuration is needed.

4.1.4.3 Tests

FCS_TLSS_EXT.1.1

304 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

305 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

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

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

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

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

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

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

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

FCS_TLSS_EXT.1.2

307 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.

High-Level Test Description
Findings: PASS

FCS_TLSS_EXT.1.3

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

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

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.
Findings: PASS

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

High-Level Test Description
Using a Lightship developed TLS client, connect to the TOE using a valid ECDHE ciphersuite and an unsupported curve and verify that the TOE fails to send back a Server Hello message and terminates the connection.
Findings: PASS

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

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

310 Test 3: [conditional] If RSA key establishment ciphersuites are supported, the evaluator shall repeat this test for each RSA key establishment key size. If any configuration is necessary, the evaluator shall configure the TOE to perform RSA key establishment using a supported key size (e.g. by loading a certificate with the appropriate key size). The evaluator shall attempt a connection using a supported

RSA key establishment ciphersuite. The evaluator shall verify (through a packet capture or instrumented client) that the TOE sends a certificate whose modulus is consistent with the configured RSA key size.

Test Not Applicable:	RSA-based ciphersuites are not supported by the TOE, hence the RSA certificate modulus (2048 bits) does not have to be tested.

FCS_TLSS_EXT.1.4

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

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

a) The client sends a Client Hello with a zero-length session identifier and with a SessionTicket extension containing a zero-length ticket.
b) The client verifies the server does not send a NewSessionTicket handshake message (at any point in the handshake).
c) The client verifies the Server Hello message contains a zero-length session identifier or passes the following steps:
Note: The following steps are only performed if the ServerHello message contains a non-zero length SessionID.
d) The client completes the TLS handshake and captures the SessionID from the ServerHello.
e) The client sends a ClientHello containing the SessionID captured in step d). This can be done by keeping the TLS session in step d) open or start a new TLS session using the SessionID captured in step d).
f) The client verifies the TOE (1) implicitly rejects the SessionID by sending a ServerHello containing a different SessionID and by performing a full handshake (as shown in Figure 1 of RFC 4346 or RFC 5246), or (2) terminates the connection in some way that prevents the flow of application data.

NIAP TD0569: Remark: If multiple contexts are supported for session resumption, the session ID or session ticket may be obtained in one context for resumption in another context. It is possible that one or more contexts may only permit the construction of sessions to be reused in other contexts but not actually permit resumption themselves. For contexts which do not permit resumption, the evaluator is required to verify this behaviour subject to the description provided in the TSS. It is not mandated that the session establishment and session resumption share context. For example, it is acceptable for a control channel to establish and application channel to resume the session.

Test Not Applicable:	The TOE supports session resumption based on session IDs and session tickets.

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

a) The evaluator shall conduct a successful handshake and capture the TOE-generated session ID in the Server Hello message. The evaluator shall then

initiate a new TLS connection and send the previously captured session ID to show that the TOE resumed the previous session by responding with ServerHello containing the same SessionID immediately followed by ChangeCipherSpec and Finished messages (as shown in Figure 2 of RFC 4346 or RFC 5246).

b) The evaluator shall initiate a handshake and capture the TOE-generated session ID in the Server Hello message. The evaluator shall then, within the same handshake, generate or force an unencrypted fatal Alert message immediately before the client would otherwise send its ChangeCipherSpec message thereby disrupting the handshake. The evaluator shall then initiate a new Client Hello using the previously captured session ID, and verify that the server (1) implicitly rejects the session ID by sending a ServerHello containing a different SessionID and performing a full handshake (as shown in figure 1 of RFC 4346 or RFC 5246), or (2) terminates the connection in some way that prevents the flow of application data.

NIAP TD0569: Remark: If multiple contexts are supported for session resumption, for each of the above test cases, the session ID may be obtained in one context for resumption in another context. There is no requirement that the session ID be obtained and replayed within the same context subject to the description provided in the TSS. All contexts that can reuse a session ID constructed in another context must be tested. It is not mandated that the session establishment and session resumption share context. For example, it is acceptable for a control channel to establish and application channel to resume the session.

High-Level Test Description
Verify that the TOE will handle session resumption via Session IDs. Verify that the TOE rejects session IDs which are not valid using a session ID which was not correctly generated.
Findings: PASS

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

a) **NIAP TD 556 -** The evaluator shall permit a successful TLS handshake to occur in which a session ticket is exchanged with the non-TOE client. The evaluator shall then attempt to correctly reuse the previous session by sending the session ticket in the ClientHello. The evaluator shall confirm that the TOE responds with an abbreviated handshake described in section 3.1 of RFC 5077 and illustrated with an example in figure 2. Of particular note: if the server successfully verifies the client's ticket, then it may renew the ticket by including a NewSessionTicket handshake message after the ServerHello in the abbreviated handshake (which is shown in figure 2). This is not required, however as further clarified in section 3.3 of RFC 5077.

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

NIAP TD0569: Remark: If multiple contexts are supported for session resumption, for each of the above test cases, the session ticket may be obtained in one context for resumption in another context. There is no requirement that the session ticket be

obtained and replayed within the same context subject to the description provided in the TSS. All contexts that can reuse a session ticket constructed in another context must be tested. It is not mandated that the session establishment and session resumption share context. For example, it is acceptable for a control channel to establish and application channel to resume the session.

High-Level Test Description
Verify that the TOE will handle session resumption via Session tickets. Verify that the TOE rejects session tickets which are not valid.
Findings: PASS

4.2 Identification and Authentication (FIA)

4.2.1 FIA_X509_EXT.1/Rev X.509 Certificate Validation

4.2.1.1 TSS

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

Findings:	According to [ST] section 6.3.6 the TOE performs X.509 certificate validation during TOE TLS client validation of remote FortiAnalyzer server X.509 certificates and when certificates are loaded into the TOE. Server certificates consumed by the TOE TLS client must have a 'serverAuthentication' extendedKeyUsage purpose. Certificate revocation checking is performed using a Certificate Revocation List (CRL). No claims are made for FPT_TUD_EXT.2 or FPT_TST_EXT.2 and therefore no TSS description is necessary to describe revocation checking in these contexts. In addition, section 6.3.7 of the [ST] indicates that additional extendedKeyUsage purposes not supported in the TOE (code signing or client authentication) are not checked.

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

Findings:	[ST] Section 6.3.6 indicates that revocation checking is performed whenever a certificate is being verified. This includes certificates verified by the TOE TLS client as well as when certificates are being loaded into the TOE.

4.2.1.2 Guidance Documentation

317 The evaluator shall also ensure that the guidance documentation describes where the check of validity of the certificates takes place, describes any of the rules for extendedKeyUsage fields (in FIA_X509_EXT.1.1) that are not supported by the TOE

(i.e. where the ST is therefore claiming that they are trivially satisfied) and describes how certificate revocation checking is performed and on which certificate.

Findings:	[SUPP] Page 19 “FortiAnalyzer configuration” section outlines the attributes that the oftp client certificate and CA certificates must have in order for it to pass the validity check.

4.2.1.3 Tests

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

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

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

High-Level Test Description
Create a sequence of three X.509 certificates: a root CA, an intermediate CA signed by the root CA and a leaf node certificate signed by the intermediate CA. Load the root CA into the TOE trust store. Force the TOE to connect to a TLS server that sends back a certificate chain in the Server Certificate message and show that the connection is accepted. Remove the intermediate CA from the TOE trust store. Force the TOE to connect to a TLS server that is configured to send the intermediate as well as the leaf and show that the connection is accepted. With the intermediate CA still missing from the TOE trust store force the TOE to connect to a TLS server that only sends the leaf certificate 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.
Findings: PASS

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

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

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

High-Level Test Description
Load the CA into the TOE trust store. Create a CRL signing certificate using a known good CA certificate that has the CRLSigning extendedKeyUsage flag enabled. Clone the known good CA certificate and **remove** the CRLSigning extendedKeyUsage. The CRL signature only depends on the (cloned) private key of the CA used to sign it and the TOE does not engage in any certificate pinning. Replace the old CA with the newly cloned CA. Verify the connection now fails due to the CRL response being signed by a CA without the proper flag.
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.
Findings: PASS

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

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

319 **NIAP TD 527 -** The following tests are run when a minimum certificate path length of three certificates is implemented.

320 **TD0527 -**Test 8: (Conditional on support for EC certificates as indicated in FCS_COP.1/SigGen). The evaluator shall conduct the following tests:

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

Note	These activities are performed in conjunction with the testing of FCS_TLSC_EXT.1.1.

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

High-Level Test Description
Construct a chain of three ECDSA certificates: a leaf, an intermediate CA and a trust anchor. Show that the leaf and chain are valid. Create a clone of the Intermediate CA, such that the public key is explicitly defined rather than being a named curve. Show that the leaf and chain are not validated correctly.
Findings: PASS

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

High-Level Test Description
Attempt to load a CA certificate with explicit format version of the Elliptic Curve parameters created in the previous test.
Findings: PASS

321 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.

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

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

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

High-Level Test Description
Load a known-good CA into the TOE trust store. Verify that connecting to our test server will yield a successful result. Clone the known good CA certificate and remove the basicConstraints extension. Replace the existing known-good CA with the cloned CA. Verify the connection fails.
Findings: PASS

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

High-Level Test Description
Clone the known good 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 connection fails.
Findings: PASS

324 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:	These tests were completed as required.

4.2.2 FIA_X509_EXT.2 X.509 Certificate Authentication

4.2.2.1 TSS

325 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:	[ST] Section 6.3.6 provides the matching algorithm used by the TOE. The TOE uses the X.509 certificate subject name and issuer name to identify certificate chains (i.e., “Issuer name of X matches the subject name of X+1”).

326 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:	[ST] Section 6.3.7 - As part of the verification process, CRL is used to determine whether the certificate is revoked or not. If the CRL cannot be obtained, then the TOE will accept the certificate in this case. There are no distinctions made between any trusted channels (there is only the one trusted channel).

4.2.2.2 Guidance Documentation

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

Findings:	[ADMIN] Page 186 “Certificates” section details how to load certificates into the TOE. [CLI] page 59 “certificate” section details the CLI commands used to import certificates. [SUPP] Page 13 “Install CA Certificate” section details how to load CA certificates into the TOE and warns about a potential warning that could arise which can be ignored.

4.2.2.3 Tests

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

329 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
Disable the CRL distribution point. Verify that the CRL cannot be fetched, and the TOE accepts the certificate.
Findings: PASS

4.2.3 FIA_X509_EXT.3 Extended: X509 Certificate Requests

4.2.3.1 TSS

330 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 TOE does not claim “device-specific information”.

4.2.3.2 Guidance Documentation

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

Findings:	[ADMIN] Page 187 “Creating a local certificate” section contains information on creating a new CSR. This section details the various fields that can be used and what each field is allowed to contain.

4.2.3.3 Tests

331 The evaluator shall perform the following tests:

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

High-Level Test Description
Using the TOE CSR generator, create a new CSR and download to an external CA entity for signing. Using OpenSSL, verify that the information in the CSR is as expected.
Findings: PASS

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

High-Level Test Description
The CSR from the previous test is signed and reimported into the TOE. The certificate is then assigned a purpose, at which point the certificate is validated. If it cannot be validated, it cannot be assigned a purpose and therefore cannot be used.
Findings: PASS

4.3 Security management (FMT)

4.3.1 FMT_MOF.1/Functions Management of security functions behaviour

4.3.1.1 TSS

332 For distributed TOEs see chapter 2.4.1.1.

Findings:	The TOE is not a distributed TOE.

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

Findings:	[ST] Section 6.4.2 - The TOE restricts the ability to modify (enable/disable) transmission of audit records to an external audit server (another FortiAnalyzer) to Security Administrators.

4.3.1.2 Guidance Documentation

334 For distributed TOEs see chapter 2.4.1.2.

Findings:	The TOE is not a distributed TOE.

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

Findings:	[SUPP] Page 19 “FortiAnalyzer configuration” section contains details on how to transmit audit logs to an external FortiAnalyzer. [CLI] Page 81 “locallog fortianalyzer (fortianalyzer2, fortianalyzer3) setting” section also contains additional details on the CLI commands available to configure the audit log offloading. [ADMIN] Page 192 “Configuring log forwarding” also shows how to configure audit log offloading from the GUI. No configuration is necessary for audit functionality when Local Audit Storage Space is full since the only claimed option is delete oldest record which is enabled by default.

4.3.1.3 Tests

336 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
Using the ‘testuser’ user (a read-only user), attempt to change the port and IP address of a FortiAnalyzer target and show that the attempt is unsuccessful.
Findings: PASS

337 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.

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

High-Level Test Description
Using the privileged ‘admin’ user, modify the IP and port of the syslog provider. Verify that the TOE attempts to communicate using the new parameters.
Findings: PASS

339 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.

340 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.

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

Note	The TOE does not claim this functionality.

342 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.

343 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.

344 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.

345 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.

346 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.

4.3.2 FMT_MTD.1/CryptoKeys Management of TSF Data

4.3.2.1 TSS

347 For distributed TOEs see chapter 2.4.1.1.

Findings:	The TOE is not a distributed TOE.

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

Findings:	[ST] Section 6.4.5 - The TOE restricts the ability to modify, delete, generate, import, or otherwise manage SSH keys, TLS and any configured X.509 certificates or private keys to Security Administrators.

4.3.2.2 Guidance Documentation

349 For distributed TOEs see chapter 2.4.1.2.

Findings:	The TOE is not a distributed TOE.

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

Findings:	[ADMIN] page 186 “Certificates” section provides information on generating, importing, modifying and deleting x509 keys. [CLI] page 46 “admin user” provides information on importing, modifying and deleting ssh public keys. [SUPP] page 12 “Key Zeroization” also provides information on key zeroization in general.

4.3.2.3 Tests

351 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.

352 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
Login as a non-admin user and attempt to create new certificates and ensure the attempt fails.
Findings: PASS

5 Evaluation Activities for Security Assurance Requirements

5.1 ASE: Security Target

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

Findings:	See above sections.

354 For distributed TOEs only the SFRs classified as 'all' have to be fulfilled by all TOE parts. The SFRs classified as 'One' or 'Feature Dependent' only have to be fulfilled by either one or some TOE parts, respectively. To make sure that the distributed TOE as a whole fulfils all the SFRs the following actions for ASE_TSS.1 have to be performed as part of ASE_TSS.1.1E.

ASE_TSS.1 element	Evaluator Action
ASE_TSS.1.1C	The evaluator shall examine the TSS to determine that it is clear which TOE components contribute to each SFR or how the components combine to meet each SFR. The evaluator shall verify the sufficiency to fulfil the related SFRs. This includes checking that the TOE as a whole fully covers all SFRs and that all functionality that is required to be audited is in fact audited regardless of the component that carries it out.

Findings:	The TOE is not a distributed TOE.

5.2 ADV: Development

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

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

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

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

359 5.2.1.1 Evaluation Activity: The evaluator shall examine the interface documentation to ensure it describes the purpose and method of use for each TSFI that is identified as being security relevant.

Findings:	From section 7.2.1 of the NDcPP : "For this cPP, the Evaluation Activities for this family focus on understanding the interfaces presented in the TSS in response to the functional requirements and the interfaces presented in the AGD documentation." The [ST] and the AGD comprise the functional specification. If the test in [SD] cannot be completed because the [ST] or the AGD is incomplete, then the functional specification is not complete and observations are required. During the evaluator's use of the product and its interfaces (the Web GUI, SSH CLI, local serial port), there were no areas that were deficient.

360 5.2.1.2 Evaluation Activity: The evaluator shall check the interface documentation to ensure it identifies and describes the parameters for each TSFI that is identified as being security relevant.

Findings:	See comments in the previous work unit.

361 5.2.1.3 Evaluation Activity: The evaluator shall examine the interface documentation to develop a mapping of the interfaces to SFRs.

Findings:	See comments in the previous work unit.

5.3 AGD: Guidance

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

363 5.3.1.1 Evaluation Activity: The evaluator shall ensure the Operational guidance documentation is distributed to Security Administrators and users (as appropriate) as part of the TOE, so that there is a reasonable guarantee that administrators and users are aware of the existence and role of the documentation in establishing and maintaining the evaluated configuration.

Findings:	The documentation is available for public download from Fortinet's documentation web site (https://docs.fortinet.com).

364 5.3.1.2 Evaluation Activity: The evaluator shall ensure that the Operational guidance is provided for every Operational Environment that the product supports as claimed in the Security Target and shall adequately address all platforms claimed for the TOE in the Security Target.

Findings:	There is only one operational environment claimed in the [ST]. All TOE platforms claimed in [ST] are covered by the operational guidance. This is evidenced by the platform equivalency.

365 5.3.1.3 Evaluation Activity: The evaluator shall ensure that the Operational guidance contains instructions for configuring any cryptographic engine associated with the evaluated configuration of the TOE. It shall provide a warning to the administrator that use of other cryptographic engines was not evaluated nor tested during the CC evaluation of the TOE.

Findings:	No other cryptographic engines are available or may be used in the TOE.

366 5.3.1.4 Evaluation Activity: The evaluator shall ensure the Operational guidance makes it clear to an administrator which security functionality and interfaces have been assessed and tested by the EAs.

Findings:	The [SUPP] document covers configuration of the in-scope functionality where additional configuration might be required.

367 5.3.1.5 Evaluation Activity: In addition the evaluator shall ensure that the following requirements are also met.

a) The guidance documentation shall contain instructions for configuring any cryptographic engine associated with the evaluated configuration of the TOE. It shall provide a warning to the administrator that use of other cryptographic engines was not evaluated nor tested during the CC evaluation of the TOE.

b) **NIAP TD 536:** The documentation must describe the process for verifying updates to the TOE for each method selected for FPT_TUD_EXT.1.3 in the Security Target. The evaluator shall verify that this process includes the following steps:

5) Instructions for obtaining the update itself. This should include instructions for making the update accessible to the TOE (e.g., placement in a specific directory).

6) Instructions for initiating the update process, as well as discerning whether the process was successful or unsuccessful. This includes instructions that describe at least one method of validating the hash/digital signature.

c) The TOE will likely contain security functionality that does not fall in the scope of evaluation under this cPP. The guidance documentation shall make it clear to an administrator which security functionality is covered by the Evaluation Activities.

Findings:	See work unit [SD] 5.3.1.3 for configuration of the cryptographic engine. The TOE claims digital signatures. The process for obtaining the update and verifying downloaded file is not corrupted is described in [SUPP]. The process for manually upgrading the TOE is provided in [SUPP] and [ADMIN].

	See work unit [SD] 5.3.1.4 for details as to what was covered by the EAs.

368 5.3.2.1 Evaluation Activity: The evaluator shall examine the Preparative procedures to ensure they include a description of how the administrator verifies that the operational environment can fulfil its role to support the security functionality (including the requirements of the Security Objectives for the Operational Environment specified in the Security Target).

Findings:	Please refer to work unit AGD_OPE.1-6.

369 5.3.2.2 Evaluation Activity: The evaluator shall examine the Preparative procedures to ensure they are provided for every Operational Environment that the product supports as claimed in the Security Target and shall adequately address all platforms claimed for the TOE in the Security Target.

Findings:	There is only one operational environment claimed in the [ST]. All TOE platforms claimed in [ST] are covered by the operational guidance. This is evidenced by the platform equivalency.

370 5.3.2.3 Evaluation Activity: The evaluator shall examine the preparative procedures to ensure they include instructions to successfully install the TSF in each Operational Environment.

Findings:	See previous work unit.

371 5.3.2.4 Evaluation Activity: The evaluator shall examine the preparative procedures to ensure they include instructions to manage the security of the TSF as a product and as a component of the larger operational environment.

Findings:	The guidance documentation provides extensive information on managing the security of the TOE as an individual product. Additional best practice guidance provided within those documents help impart a culture of secure manageability within a larger operational environment.

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

The preparative procedures must:

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

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

Findings:	The entire [SUPP] document is designed to ensure the administrator is aware of how to configure the TOE to provide a protected administrative capability. The TOE has default TOE passwords. However, when placing the device into FIPS-CC mode, the administrator is required to change the password to meet the minimum password requirements as stated in the [SUPP]. These complexity requirements are enforced by the TOE rather than by policy.

5.4 ATE: Tests

373 The focus of the testing is to confirm that the requirements specified in the SFRs are being met. Additionally, testing is performed to confirm the functionality described in the TSS, as well as the dependencies on the Operational guidance documentation is accurate.

374 The evaluator performs the CEM work units associated with the ATE_IND.1 SAR. Specific testing requirements and EAs are captured for each SFR in Sections 2, 3 and 4.

375 The evaluator should consult Appendix 709 when determining the appropriate strategy for testing multiple variations or models of the TOE that may be under evaluation.

376 Note that additional Evaluation Activities relating to evaluator testing in the case of a distributed TOE are defined in section A.9.3.1.

Findings:	The evaluator performed the CEM work units associated with ATE_IND.1 and recorded them in a report provided to the CB. The evaluator performed testing on additional models consistent with Appendix 7 and in line with the CB requirements. The TOE is not a distributed TOE.

6 Vulnerability Assessment

377 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.

378 **NIAP TD 547 -** The developer shall provide documentation identifying the list of software and hardware components that compose the TOE. Hardware components should identify at a minimum the processors used by the TOE. Software components include applications, the operating system and other major components that are independently identifiable and reusable (outside of the TOE), for example a web server, protocol or cryptographic libraries, (independently identifiable and reusable components are not limited to the list provided in the example). This additional documentation is merely a list of the name and version number of the components and will be used by the evaluators in formulating vulnerability hypotheses during their analysis.

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

379 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 and - US-CERT: http://www.kb.cert.org/vuls/html/search - Offensive Security Exploit Database: https://www.exploit-db.com/ - Rapid7 Vulnerability Database: https://www.rapid7.com/db/vulnerabilities - OpenSSL Vulnerabilities: https://www.openssl.org/news/vulnerabilities.html - Google Type 1 Hypothesis searches were conducted and included the following search terms:

- Fortinet;

- FortiAnalyzer;

- OpenSSL;

- OpenSSH;

- Apache

- Intel i3-6100 Skylake

- Intel Xeon Bronze 3106 Skylake

- Intel Xeon E5-2620v3 Haswell

- Intel Xeon E5-2630v3 Haswell

- Intel Xeon Gold 5118 Skylake

- Intel Xeon E5-2640v4 Broadwell

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

The type-2 hypotheses for the NDcPP does not apply as the TOE does not use RSA based ciphersuites.

The evaluation team developed Types 3 and 4 flaw hypotheses in accordance with Sections A.1.3, A.1.4, and A.2 (of the [SD]), and that no residual vulnerabilities exist that are exploitable by attackers with Basic Attack Potential in accordance with the guidance in the CEM.